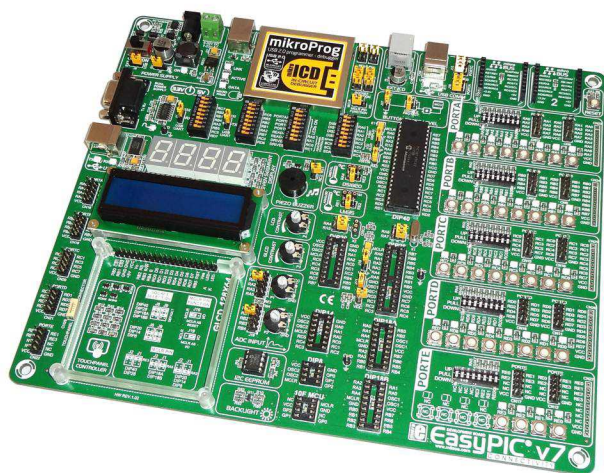
型番:PICD-700SX　　　　　REV.6.00

USB2.0対応 PICマイコンライター搭載
350種類のデバイスに対応!

PICマイコン開発用統合評価ボード

取扱説明書
お使いになる前にこの説明書をよくお読みの上正しくお使いください。

ボード本体

製品の概要

PICマイコン開発用統合評価ボード(型番:PICD-700SX、以下型式で記載)はオンボードのUSB2.0対応PICプログラマー、mikroC用ハードウエアインサーキットデバッガ(ICD)を搭載し、様々な周辺回路をワンボードに凝縮した開発・実験・試作・学習用とあらゆる用途に使用できる8ビットPICマイコン用の統合開発ボードです。

ボード上には、8ピン・14ピン・18ピン・20ピン・28ピン・40ピンのICソケットが用意されており、現在流通している250種類の8ビットPICマイコンを装着できます。PIC10F/12F/16F/18Fなどお馴染みのデバイスの他、最近登場したナノワットテクノロジーを採用した3.3V仕様のPIC18xxKシリーズやPIC18xxJシリーズのデバイスにも対応しました。1つのボードで5V系デバイスと3.3V系デバイス両方に対応しています。

どのピン数のデバイスであっても、基板上のスイッチやジャンパーソケットを操作することで、ボードに搭載されている各周辺回路を効率よく利用することができます。また全I/Oピンはボード右側に実装されているヘッダピンから取り出すことができます。

ボード上に搭載のUSB接続PICライターは、数多くのPICマイコンに対応しており、基板上に装着したデバイスにプログラムを書き込むことができます。USB接続を行った場合、USBバスからPICD-700SXの電源を取りますので、別途電源を準備する必要がありません。

コードサイズ限定版(2Kワードまで)の体験版Cコンパイラーを付属。ICD(インサーキットデバッグ)機能搭載なので手軽にC言語によるICD機能が体験できます。また、本書には簡単なC言語チュートリアルを収録していますので、すぐにC言語による開発を体験できます。

パッケージの内容

■同梱物

- ・PICD-700SXボード本体
- ・PIC18F45K22
- ・PIC16F1939　　（ボードに装着済み）
- ・8MHz水晶発振子　（ボードに装着済み）
- ・16文字×2行液晶ディスプレイモジュール　（ボードに装着済み）
- ・USBケーブル
- ・CD-ROM
- ・マニュアル(本書)

※PICD-700SXの全回路図は付属のCD-ROM内データシートフォルダに収録されています。

ドライバのインストール及びパソコンとの接続

■デバイスドライバーのインストール

PICD-700SXをパソコンと接続する前に、デバイスドライバーをパソコンにインストールします。

1 付属のCD-ROMをパソコンのCD-ROMドライブに挿入してエクスプローラー等で内容を表示します。

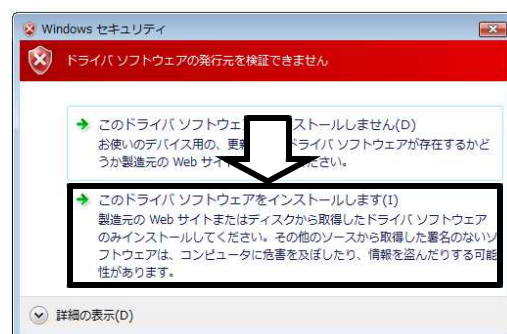
2 "USB Programmer Software"のフォルダを開きます。

3 "Driver"フォルダを開きます。

4 OSごとにフォルダがあります。ご使用のパソコンのOSにあったフォルダを開きます。
※WindowsVista,7,8に関しては32ビット版用と64ビット版用がありますので間違えないようご注意ください。

5 フォルダ内には実行ファイル、"USB18PRG～.exe"というファイルがありますのでダブルクリックして実行します。

6 実行するとダイアログが表示されますので、そのまま"次へ"をクリックして続行します。

7 ドライバーのインストールが開始されます。
WindowsVista又は7環境で下記のような警告ダイアログが表示された場合には、"このドライバソフトウエアをインストールします"をクリックして続行してください。

下図のメッセージが表示された場合には
"インストール"ボタンをクリックしてください。

※ウイルス対策ソフトウエアなどがインストールされている場合、インストール時にウイルス対策ソフトウエアが警告を表示する場合があります。その場合には、インストールを許可するように設定してください。

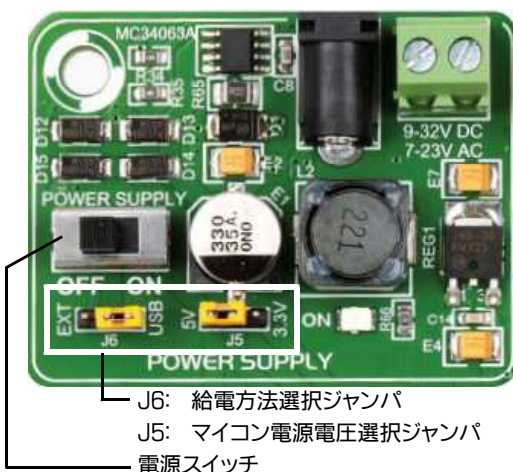
8 インストール作業が完了すると、下図のような画面になりますので、"完了"ボタンを押してインストールを完了します。

この時、"状態"のボックスに「使用できます」と表示されていることをご確認下さい。エラーが表示された場合には、ご使用OSの種類と、実行したデバイスドライバーの種類が一致しているかご確認下さい。

■パソコンと接続する

パソコンのUSBポートにPICD-700SXを接続します。
接続に際しては必ずパソコン本体のUSBポートに接続してください。
USBハブを介しての接続の場合には正しく動作しないことがあります。

1 本体左上にある電源回路部で、供給する電圧を設定します。

J6: 給電方法選択ジャンパ
J5: マイコン電源電圧選択ジャンパ
電源スイッチ

USBバスパワーで本体の電源を給電する場合には、J6の給電方法選択ジャンパを、"USB側"にセットします。これでUSBバスパワーにて本体が動作します。

ACアダプタにて別途外部から電源を給電する場合には、J6のジャンパーソケットを"EXT"側に設定します。これで外部からの給電になります。ACアダプタは径φ2.1mmのDC9V～12V出力でセンター極性がプラスのものをご利用下さい。

J5はマイコンに供給する電源の電圧を設定するジャンパーソケットです。+5Vで動作するPICマイコンの場合には"5V"側に、+3.3Vで動作するPICマイコンの場合には"3.3V"側に設定します。
工場出荷時にはPIC18F45K22が装着されています。このデバイスは5V動作ですので、"5V"の方にジャンパーソケットが装着されていることを確認します。

PIC18FxxJxx　デバイス使用時は3.3Vに!

PIC18Fシリーズで、Jが付くデバイスは電源電圧の仕様が3.3Vです。よってJが付くデバイスを使う場合にはJ5のジャンパーソケットを"3.3V"側に設定する必要があります。間違えてしまうとデバイスを破損してしまいますのでご注意ください。

2 USBケーブルを電源スイッチの近くのUSBポートに接続し、PICD-700SXをパソコンと接続します。

3 PICD-700SXの電源スイッチをONにします。

→接続するとPOWERの緑LEDが点灯します。
→新しいハードウエアの検出ウィザードが自動的に起動します。

4 Windows8の場合にてモダンUIではなく、デスクトップを表示させてデスクトップUIの環境でインストールを行ってください。すべてのソフトウエアはデスクトップUI環境で動作します。

WindowsVista、7、8の場合には、PICD-700SXを接続すると自動的にデバイスドライバーのセットアップが開始されます。
あらかじめデバイスドライバーは先の手順にてインストールされていますので、WindowsVista、7、8は自動的にデバイスドライバーを検出してインストールを完了します。

接続後しばらく待ちますと、下図のようなポップアップがタスクバーに表示されます。

上図のホップアップが表示されればインストールは完了です。

5 PICD-700SXボード上の"LINK"の黄LEDが点灯していることを確認してください。(USBコネクタのすぐ下にあるLEDです。)

黄LEDの点灯はPC側にPICD-700SXが正しく認識されていることを示します。"LINK"のLEDが点灯していない場合には、正しくドライバーがインストールされていません。再度手順を確認の上、ドライバーをインストールし直しておためしください。

"LINK" LEDが点灯していることを確認します。点灯していない場合パソコンに本機が認識されていません。

Windows8.1及び10、64ビット版で使用する場合のご注意

Windows8.1、10の64ビット版で本機のドライバーをインストールし、ボードをパソコンに接続すると、環境によっては正しくドライバーがインストールできない場合があります。

これは、Windows8.1、10ではデフォルトの設定で署名のないデバイスドライバーがシステムの設定変更を拒否するように設計されていることによります。この問題はデジタル署名のないデバイスドライバーのインストールを許可するようにWindows8.1、10の設定を変更することで回避できます。

インストール時にデバイスドライバーをインストールしたにもかかわらず、PICD-700SXをパソコン本体と接続しても"LINK LED"が点灯しない場合には、お手数ですが別紙「各種デバイスドライバーをWindows7及び8.1の64ビット版にインストールする場合のご注意」に記載の方法でデジタル署名の確認を回避する設定にしてお試し頂けますようお願いいたします。
この方法で設定を変更すれば正しくインストールが完了し、問題なくお使い頂けることを確認しております。

WIndowsVista,7,8.1,10環境でご使用のお客様へ ユーザーアカウント制御無効設定のお願い

WindowsVista、7、8、10環境で、PICD-700SXをご使用の場合、Windowsに搭載されたユーザーアカウント制御(UAC)機能がUSBポートへのアクセスを拒否することにより、下図のようなエラーメッセージが表示されて、書き込みができないという現象を確認しています。

この現象は、Windowsにおいて標準で有効になっているユーザーアカウント制御機能を無効に設定することで回避できます。下記の方法で設定を変更して頂けますようお願い致します。

■Windows Vistaのユーザーアカウント制御を無効に設定する

1 スタートボタンをクリックして、コントロールパネルを表示します。

2 コントロールパネルから"ユーザーアカウント"をダブルクリックします。"ユーザーアカウント"アイコンがない場合には、ウインドウ左上の"クラシック表示"をクリックして表示を変更します。

3 "ユーザーアカウント制御の有効化または無効化"のリンクをクリックします。

4 "ユーザーアカウント制御(UAC)を使ってコンピューターの保護に役立たせる"のチェックを外します。

5 "OK"ボタンを押して完了します。パソコンを再起動します。

■Windows7、8.1、10のユーザーアカウント制御を無効に設定する

1 コントロールパネルを表示します。

2 コントロールパネルから"ユーザーアカウントと家族のための安全設定"をクリックします。

3 "ユーザーアカウント"をクリックします。

4 "ユーザーアカウント制御設定の変更"をクリックします。

5 画面に表示されるスライドバーを一番下の"通知しない"に設定します。

6 "OK"ボタンを押して完了します。パソコンを再起動します。

ソフトウエアのインストール

PICマイコンの開発に必要なソフトウエアをインストールします。
CD-ROMをパソコンに挿入し下記の手順でインストールしてください。

■2Kワード限定版mikroC PRO for PIC

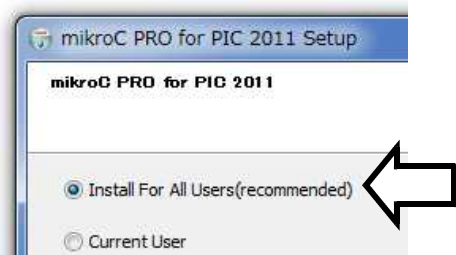
CD-ROMには、PIC12F/16F/18Fシリーズの8ビットPICマイコン用、ANSI規格準拠のCコンパイラー、mikroC PRO for PIC(以下mikroCと記載)が収録されています。mikroCは、豊富な組込関数を搭載している他、分かりやすいインターフェイスで初心者の方でもすぐに使える8ビットPIC用のCコンパイラーです。

本PICD-700SXには、生成できるHEXファイルのコードが2Kワードまでにサイズ制限されている体験版が収録されています。体験版と言えども、コードサイズに制限がある以外は製品版と同等の機能が利用できます。
利用した後、気に入った場合には、コードサイズに制限のない製品版を当方からPICD-700SXお買上のお客様特別価格(販売価格の10%引き)にてお買い求め頂けます。なお本書に記載のチュートリアルはすべて本mikroCでの開発例を掲載しております。

Cコンパイラーを使用しない場合には、インストールしなくてもかまいませんが、インストールしない場合には、次の「PICプログラマー用ソフトウエア」を必ずインストールしてください。

1 CD-ROM内の"Cコンパイラー"フォルダを開きます。
"SETUP.EXE"をダブルクリックして実行します。
"Next>"をクリックして続行します。

2 "License Agreement"のダイアログが表示されたら内容をよくご確認の上、同意する場合には"I accept the terms in the License Agreement"にチェックを入れて、"Next>"クリックして続行します。
なお同意しないとインストールは続行できません。

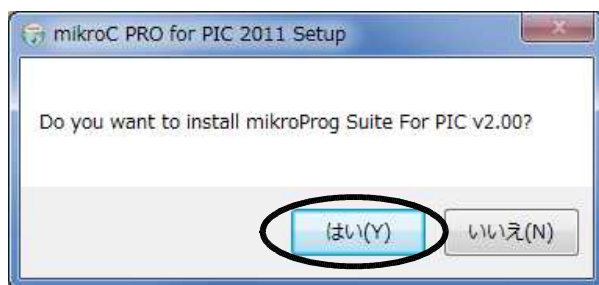
3 次の画面では、"Install For All Users(recommended")にチェックを入れて、"Next>"ボタンを押して続行します。

4 インストールするソフトウエアコンポーネントを選択します。下図の通り、3つすべての項目にチェックが入っていることを確認します。

(CompilerとHelp Filesはグレーアウトしていますが、そのまま続行してください。)
"Next>"ボタンを押してください。

5 インストールディレクトリを選択します。
通常はそのままの設定でかまいませんが、変更する場合には、"Browse"ボタンをクリックして、インストール場所を指定してください。
なお、インストール場所にサンプルプログラムなどもインストールされますので、インストールした場所はメモを取るなどして覚えておいてください。

※ディレクトリ名に日本語や2バイト文字が含まれないようにしてください。

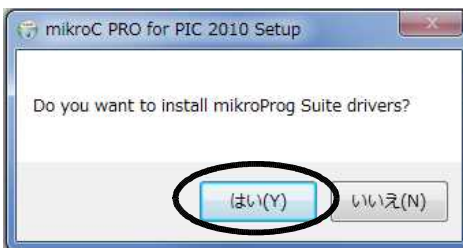
設定が完了したら、"Install"ボタンをクリックします。
→インストールが開始されます。

6 インストールが完了すると、"Finish"ボタンが表示されますので、クリックしてインストールを完了します。
引き続き、下図のようなメッセージが表示されますので、"はい"をクリックします。

書き込みソフトウエアをインストールします。インストールウィザードが起動しますので、"Next>"をクリックして続行します。

→ライセンス許諾画面が表示されますので、内容をお読みになって、同意される場合には、"I accept the terms in the License Agreement"にチェックを入れて、"Next>"をクリックします。

※同意しないとインストールは続行できません。

7 続いて、インストールするディレクトリを選択するダイアログが表示されます。通常はそのままの設定でかまいませんが、インストールするディレクトリを変更場合には、"Browse"ボタンを押して、インストール場所を変更してください。

8 インストール場所を設定したら、"Install"ボタンを押してインストールを続行します。
インストールが完了したら"Finish"ボタンを押して、インストールを終了します。
続いて"Do you want to install mikroProg Suite drivers?"というメッセージが表示されますので、"はい"をクリックします。

9 これで、ソフトウエアのインストールは完了です。

■PICプログラマ用のソフトウエア(mikroProg Suite)

PICへプログラムを書き込む際に使用するプログラマソフトウエア、mikroProg Suiteのインストールを行います。mikroCをインストールしなかった場合のみインストールしてください。mirkoCをインストールした場合には、一緒に書き込みソフトウエアもインストールされますので、この項目は読み飛ばしてください。

【必ずお読み下さい】

mikroCをご使用になる場合には、mikroCを先にインストールして頂き、ここで説明しているmikroProg Suiteのインストールはしないでください。ソフトウエアが二重にインストールされてしまい、不具合の原因になる場合があります。mikroCには、mikroProg Suiteが含まれており、mikroCインストール時に自動的にmikroProg Suiteがインストールされるためです。

1 CD-ROM内の"USB Programmer Software"内にある"SETUP.EXE"をダブルクリックして実行します。インストーラーが起動しますので、画面の指示に従ってインストールしてください。

2 "インストールは完了しました"ダイアログが表示されたら,「完了」をクリックして終了します。

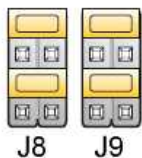
使用するデバイスに応じた設定

ボードを使用する前に、使用するデバイスに応じた設定をする必要があります。設定は主にピン数に合わせた設定と、使用するデバイスの電源電圧の設定です。設定が間違っているとデバイスを破損したり、正しくデバイスが動作しないことがありますので必ずご確認ください。

①ピン数に応じた設定 (J8,J9)

PICD-700SXでは本体のソケットに装着するPICのピン数に応じてジャンパーピンJ8及びJ9にて設定を行う必要があります。
使用するデバイスのピン数に応じて、使用前に必ず下記の通り設定を行ってください。

●20、14、8ピンのデバイスを使用する場合

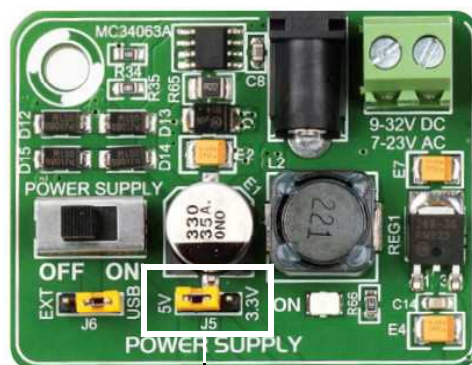
●40、28、18ピンのデバイスを使用する場合

②ピン数に応じた設定 (J1,J2)

ピン数に応じて、MCLRピンの位置をジャンパーJ1とJ2で設定する必要があります。MCLRピンはハードウエアリセットピンで通常時はVdd電位、リセットをかけたい場合だけGNDレベルにします。

使うデバイスのピン数に応じて、次のように設定してください。

標準で付属しているPIC18F45K22及びPIC16F887はいずれも40ピンデバイスですので、上図左のJ1、J2とも一番上になるように設定されています。

③電源電圧に応じた設定

PICD-700SXでは、+5.0Vで動作するデバイスと+3.3Vで動作するデバイス両方に対応しています。使用するデバイスの電源電圧に応じて、J5ジャンパーにて設定します。

J5ジャンパー　左側が5V、右側が3.3V

電源電圧については、使用するデバイスのデータシートに記載されておりますので、使用する前には必ずご確認ください。基本的に下記のようになっています。

●5V動作のデバイス
PIC10F, PIC12F, PIC16F, PIC18F(Jが付かないもの)

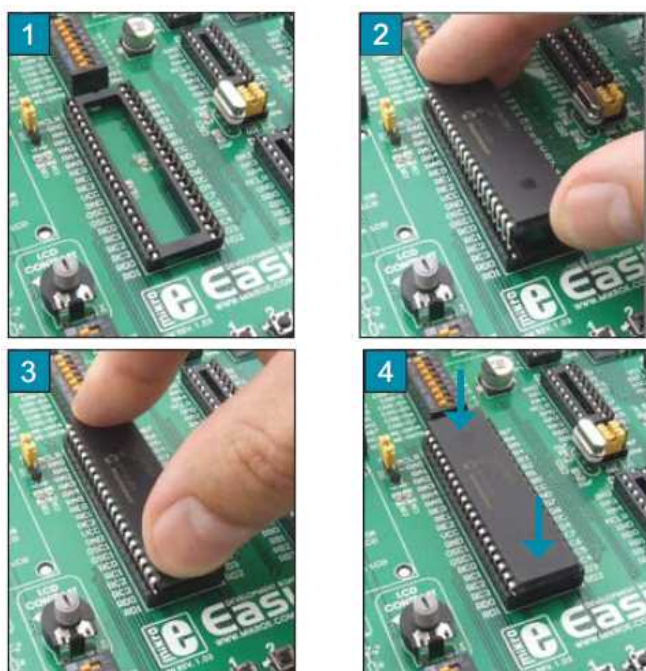
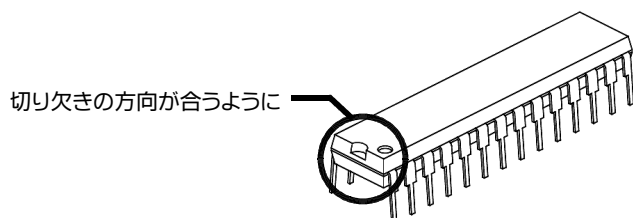
●3.3V動作のデバイス
PIC18FxxJ (Jが付くデバイス)

※付属のPIC16F1939や、PIC18F45K22はいずれも"J"の付かない5V動作のデバイスです。

PICマイコンの装着

使用するPICマイコンを、PICD-700SXボード上のICソケットに装着します。工場出荷時は、PIC16F1939がすでに装着されています。

装着できるPICマイコンは、必ず1種類だけです。複数のソケットに複数のPICマイコンを装着すると故障の原因となります。
装着時には装着方向を間違わないようにICソケットの切り欠き部分とPICマイコンの切り欠き部分が合うように正しく装着してください。

切り欠きの方向が合うように

※デバイスを取り外す場合には、ICエクストラクター等の器具をご利用下さい。脱着の際にはICやICソケットの損傷にご注意ください。

■8ピンPICマイコンの装着
8ピンのPICマイコンにはPIC10FシリーズとPIC12Fシリーズがあります。PICD-700SXは両方のデバイスに対応していますが、ソケットが異なります。
PIC10Fシリーズの場合には、"10F MCU"と記載されたソケットに装着します。PIC12Fシリーズの場合には、"DIP8"と記載されたソケットに装着します。

■18ピンPICマイコンの装着
18ピンデバイス用のソケットは、"DIP18A"と"DIP18B"と2種類あります。これは、現在流通している18ピンのPICマイコンでは2種類のピン配置のものがあるためです。例えば18ピンデバイスでもPIC16F628Aと、PIC18F1220ではピンのアサインが違っています。
18ピンデバイスを使用する際には、ご使用になるPICマイコンのデータシートでピン名称と配置をご確認頂き、基板上のシルク印刷と実際のデバイスのピン名称が一致するソケットの方にデバイスを装着してください。
※PICマイコンの1ピンがRA0か、RA2なのかで判断できます。

■その他のピン配置が特殊なPICマイコンへの対応
多くのPICマイコンではピン配置に互換性があります(ピンコンパチブル)が、一部のデバイスはピン配置が特殊になっているため、ジャンパー設定で設定を行う必要な物があります。特に、USB機能のついているデバイスでは、Vcapピンがありそれらを設定する必要があります。

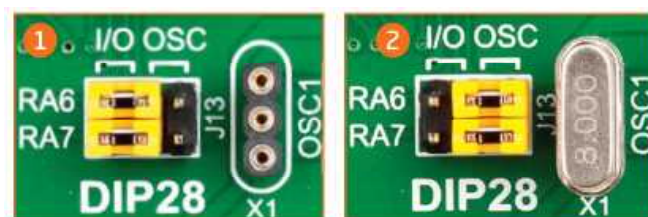
次の表のデバイスを使用する場合には、表の内容に従ってジャンパー設定を設定してご使用ください。
表に記載されている以外のデバイスを使用する場合には、設定を必ず元の状態に戻してください。

デバイス名	ジャンパ番号	設定位置
PIC16F724 PIC16F727	J22	"Vcap"側に設定
PIC18F44J10 PIC18F45J10	J17	"Vcap"側に設定
PIC18F24J10 PIC18F25J10 PIC18F2xJ50 PIC18F2xJ11	J10	"Vcap"側に設定
PIC16F722 PIC16F723 PIC16F726	J23	"Vcap"側に設定
PIC18F2331 PIC18F2431	J20	ソケットを装着 ※RA5がVccに接続

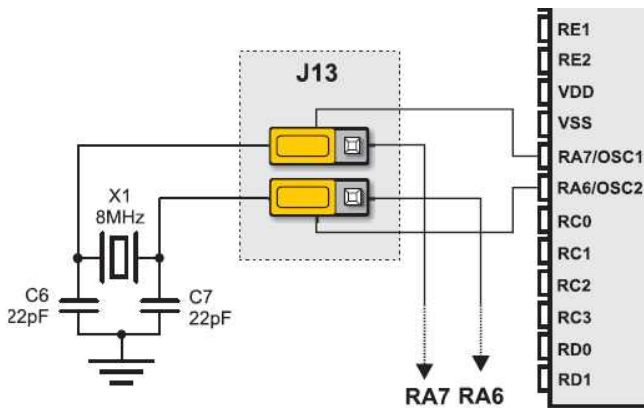
外部発振子の取り付けと、内部／外部発振子の選択

■外部発振子の取り付け
PICD-700SXでは、PICマイコンに動作クロックを供給する外部発振子を取り付けることができるように2つの発振子用ソケットが準備されています。必要に応じて、水晶発振子又はセラミック発振子を取り付けてご使用になれます。
基板上には、"OSC1"と"OSC2"があり"OSC1"は40ピン・28ピン・18ピンデバイス用のソケット、"OSC2"は20ピン・14ピン・8ピンデバイス用のソケットです。

■内部／外部発振子の選択
PICマイコンの種類によっては、発振子を内蔵しているものがあります。PICD-700SXでは、外部発振子を使用するか、内部発振子を使用するかをジャンパーソケットJ13及びJ14で設定します。
この設定が正しくないとPICマイコンは動作しませんので、使用前によくご確認下さい。なお、内蔵発振子を使用する場合には適宜プログラム内でOSCCONレジスタなど、内蔵発振子を使うための設定を記述する必要があります。詳しくはデバイスのデータシートをご参照ください。

OSC1のエリアはJ13で、OSC2のエリアはJ14で設定します。
外部に水晶発振子又はセラミック発振子を使用する場合には、J13又はJ14のジャンパーソケットを"OSC"と印刷された方(左側)にセットします。ソケットには発振子を装着します。下図はJ13の例です。

内蔵発振子使用時の設定

外部発振子使用時の設定

内部結線は下図のようになっています。

電源の投入方法

PICD-700SXは、USB接続にてUSBバスから電源の供給を受けることができます。この場合にはPICD-700SXには別途電源を供給する必要はありません。

PICD-700SX側で多くの電流(およそ300mA以上)を消費する予定がある場合にはUSBバスパワー給電では電力が不足する場合があります。この場合には別途外部からACアダプタにて電源を供給する必要があります。

給電方法はジャンパーソケットJ6によって設定を行います。
なお工場出荷時の設定はUSBバスパワー給電の設定になっています。

■USBバスパワーから給電する場合
PICD-700SXボード上のジャンパーピン(J6)を「USB」側にショートします。

■ACアダプタから給電する場合
PICD-700SXボード上の"ジャンパーピン(J6)を「EXT」間にショートします。

MCLRピンの使用に関する設定

最近のデバイスでは、MCLRピンをマイコンの外部リセット用のピンとしてではなく、入力ピン(又は入出力ピン)として利用できるデバイスが増えています。MCLRピンをどちらの設定にするかは、HEXファイルをマイコンに書き込む際に設定するコンフィギュレーションビットで設定を行いますが、ボード上でもジャンパーソケットの設定が必要です。設定はJ19ジャンパーソケットで行います。

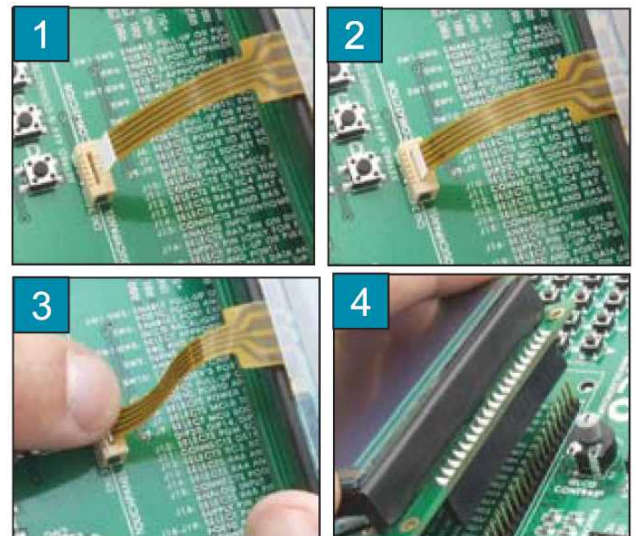
J19ソケットを"I/O"と書かれた方に設定すると、MCLRピンは入出力ピンとして利用できます。この設定では基板上のリセットボタンは無効となります。

J19ソケットを"MCLR"と書かれた方に設定すると、MCLRピンはハードウエアリセットのピンとなります。この設定にすると、基板上のリセットボタンを押すことでリセットをかけることができます。

GLCDを取り付ける場合

PICD-700SXには、128×64ドットのグラフィックLCDを取り付けることができます。また4線式抵抗膜方式のタッチパネルを搭載したGLCDを取り付ければ、基板上のフラットケーブルコネクタにタッチパネルの信号線を取り付け、タッチパネルコントローラに信号を接続することができます。

1 GLCDを取り付ける場合には、必ずPICD-700SXの電源を切断してください。

2 タッチパネル付きのGLCDの場合には、GLCDを取り付ける前に、タッチパネルのフラットケーブルを基板上のコネクタに押し込んで装着します。コネクタは抜けないように堅い部分をコネクタに押し込んでください。

3 続いて128×64ドットGLCDをソケットに真っ直ぐ差し込みます。

※GLCDの脱着時は必ずPICD-700SXの電源を切断した上で作業を行ってください。

USB-UART変換IC搭載実験用USBポートを使う場合

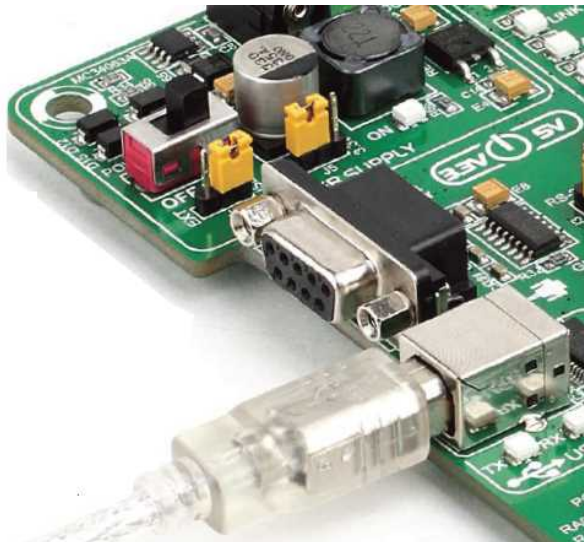
PICD-700SXでは、RS232Cポートの他にRS232Cポートがないパソコンでも非同期式シリアル通信(UART)ができるように、USB-UART変換IC(FTDI社製、FT232R)を用いたUSBポートを搭載しています。最近のパソコンではRS232Cポートを搭載しないものが多いため、USB経由でUART(非同期式シリアル通信)を実験して頂けるようになりました。

PICマイコン側は、通常のUARTを使う場合と同じプログラムで動作します。パソコン側は、仮想COMポートドライバーをインストールすることでCOMポートが作られ、そのポートに従来のRS232Cと同様にターミナルソフトでアクセスすることでRS232C通信が行えます。

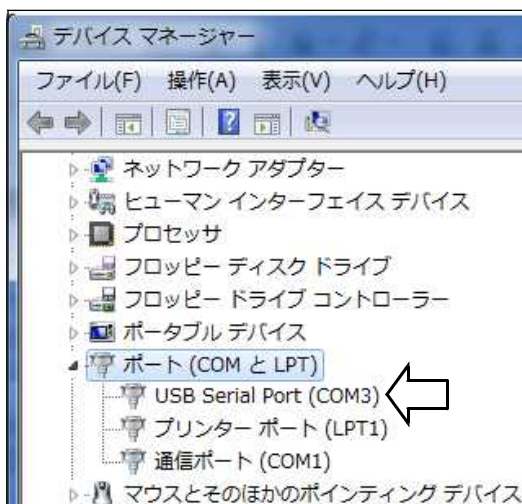
また、PICBASIC PRO Compiler用のICD機能付き統合開発環境、MicroCode Studio Plus(MCSP)でICD機能を使う場合でも、このUSBポートを使用してパソコンと通信させることができます。

ご使用にあたっては、パソコン側に仮想COMポートドライバーのインストールが必要となりますので下記の手順でインストールをします。

1　PICD-700SXの電源を投入します。
USBバスパワーでもACアダプタからの給電でもどちらでもかまいません。

2　PICD-700SX本体左側のUSB-UARTと書かれたUSBポートにUSBケーブルを挿入してパソコンと接続します。
※USBケーブルは別途ご用意下さい。

3　パソコンで新しいハードウエアデバイスが認識され、インストールが開始されます。

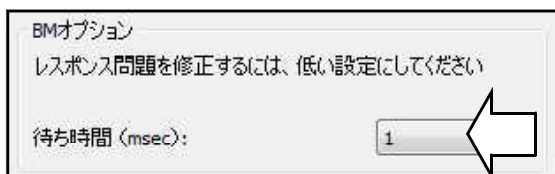
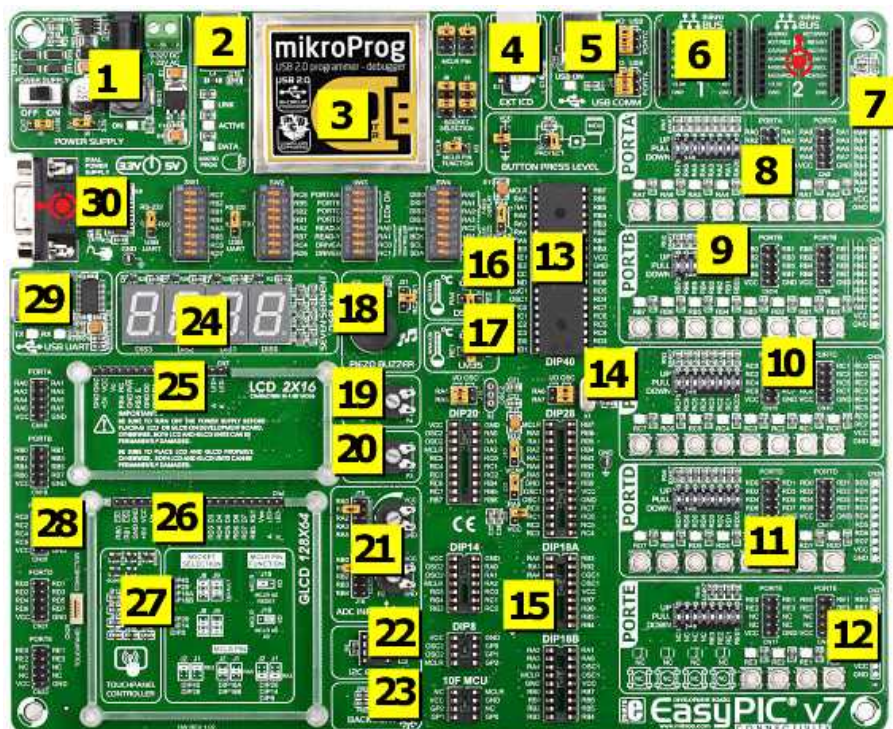
4　ドライバーの場所を尋ねるメッセージが表示された場合には、CD-ROM内の "FT232Rドライバー" フォルダーを指定します。ドライバーのインストールが行われます。
なおドライバーのインストールは2回行われます。（1回目がFT232Rチップドライバーのインストール、2回目が仮想COMポートドライバーのインストールです。）

5　インストールが完了すると、パソコンに仮想COMポートが作られます。このポートにアクセスすることでPICD-700SXに搭載のPICマイコンのUARTピンにアクセスできます。
仮想COMポートの番号確認は、デバイスマネージャより行います。
Windowsのデバイスマネージャを開き、"ポート(COMとLPT)"のツリーを展開すると、"USB Serial Port(COMx)"という表示がありますので、ここで表示された(COMx)の番号がCOMポートの番号となります。下図例ではCOM3になっています。

6　このインストール直後の状態で普通に使うことができますが、一部のアプリケーションでレイテンシーの設定を変更しないと不具合が起こることがあります。次の手順でレイテンシーの設定を変更してご使用下さい。
デバイスマネージャにおいて、インストールされた"USB Serial Port(COMx)"の項目を右クリックして"プロパティ"を選択します。

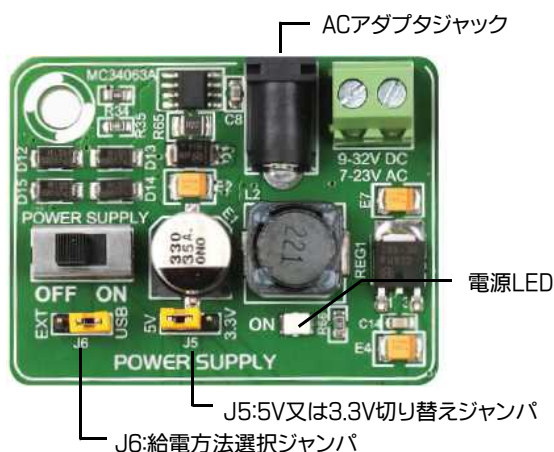
7　プロパティ画面が表示されますので、"ポートの設定"タブに移動して、"詳細設定"ボタンをクリックします。

8　画面の中から"BMオプション"の項目の"待ち時間(msec)"のプルダウンをクリックして、値を "1"に設定してください。

"OK"ボタンを押して完了します。

PICD-700SXの各部の説明

①電源回路部／ACアダプタジャック／電源スイッチ

PICD-700SXの電源回路部です。USBからのバスパワー給電又は、ACアダプタで本体に給電をします。ACアダプタ使用時は、DC9V～12V出力で径2.1φで500mA以上が取り出せるものをご使用下さい。

USBバスパワー給電か、ACアダプタ給電かをJ6ジャンパーで設定できます。USBバスパワー給電の場合には、"USB"側に、ACアダプタ給電の場合には"EXT"側にジャンパーソケットをセットしてください。

PICD-700SXは電源電圧が+5Vのデバイスと、+3.3Vのデバイス両方を1つのボードで使うことができます。電源電圧はJ5で設定ができます。使用するデバイスの仕様に合わせて設定を行ってください。詳しくは本書6ページをご参照ください。

電源が投入され、スイッチをONにすると電源LEDが点灯します。

②USBマイコンライター用USBポート

パソコンとPICD-700SXを接続する時に接続するUSBタイプBポートです。
その下のLEDは次のようになっています。

LINK LEDは、パソコンに本ボードが認識されると点灯します。

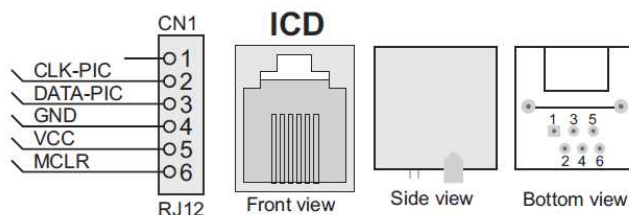
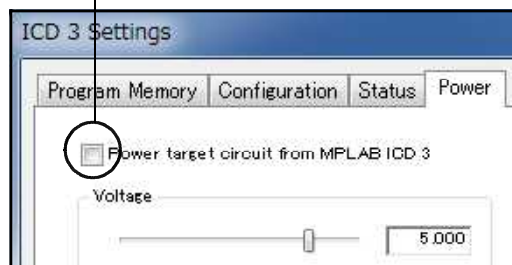
ACTIVE LEDは、PICライターが動作している時点灯します。

DATA LEDは、PICマイコンにデータが転送されている時点灯します。

③マイコンライター／mikroC用ICD回路部

約250種類のPICマイコンに対応するマイコンライターと、付属のCコンパイラ、mikroCでインサーキットでバッグ(ICD)を実行するために必要な回路です。

④ICD接続用モジュラージャック

マイクロチップ社純正のインサーキットデバッガ(MPLAB-ICD)を接続するためのモジュラージャックです。本ジャックにMPLAB-ICDを接続することで、本ボードをターゲットボードとして使用することができます。ピンアサインは下図の通りです。

MPLAB-ICDでは、ターゲットボードの電源についてMPLAB-ICD側から給電する方法と、ターゲットボードに別途電源を給電する自己給電のどちらかを選ぶ機能がありますが、必ずターゲットボードは自己給電を設定して行って下さい。設定は、MPLAB IDE内の"Programmer"又は"Debugger"メニュー内の"Settings"から"Power"タブで、"Power Target circuit from MPLAB ICD3"のチェックを外してください。

MPLAB-ICD接続前に必ず、上の設定をしてください。
また、MPLAB-ICDでプログラムの書き込みやICDを実行する場合には、使用しているデバイスのPGC線及びPGD線はプルアップやプルダウンはしないようにご注意ください。

※MPLAB-ICDの使い方はICDのマニュアルをご覧ください。

⑤実験用USBポート

USB通信機能搭載デバイスでUSBの実験を行う際に使用できるUSBポートです。USB搭載デバイスでUSB通信を行う場合には、J12又はJ18のジャンパーソケット3つ全てを右側の"USB"側にセットします。USB通信機能を使用しない場合には、3つ全てを左側の"I/O"側にセットします。

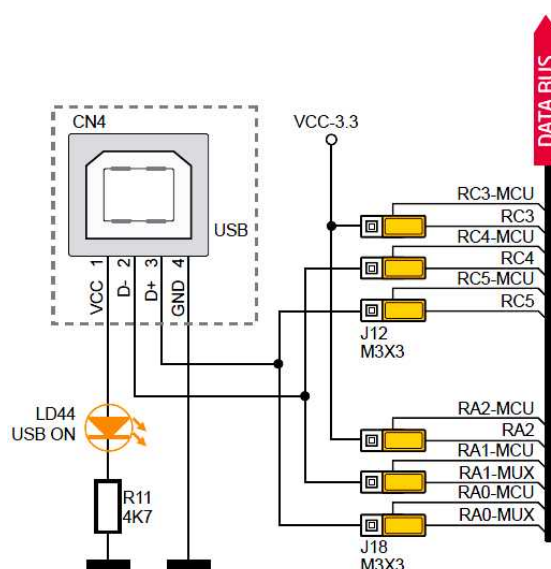
PICマイコンの種類によってUSBデータ線のD+とD-線がPORTCにアサインされているものと、PORTAにアサインされているものがあります。PORTCにアサインされているデバイスを使う場合にはJ12を、PORTAにアサインされているデバイスを使う場合にはJ18のジャンパーソケットを設定してください。使用するデバイスのデータシートをご覧になって、どのポートを使っているかご確認ください。
(J12とJ18両方を同時にUSB側にセットすることはありません。)

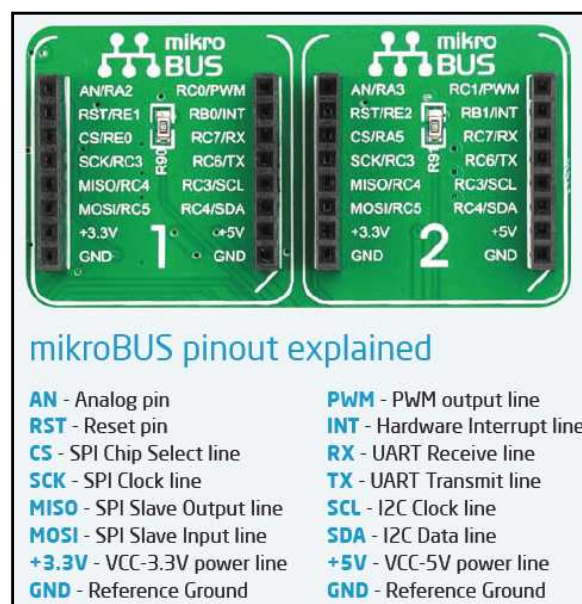
USBを使わない時

PORTCをUSBとして使用する時

PORTAをUSBとして使用する時

主にPIC18F1xK50シリーズの場合にはJ18側を、
PIC18Fxx(J)50・PIC18Fxx(J)53・PIC18Fxx(J)55・PIC18Fxx58の場合にはJ12側を"USB"位置に設定します。

※本実験用のUSBポートからPICD-700SXの電源は供給できません。よってUSBポートの実験をする場合には、PICD-700SXにはACアダプタから電源を給電するか、パソコンにUSBポートが2つあれば、USBライターと実験用USBポートと2つのポートを接続することになります。

⑥mikroBUSコネクタB部

将来拡張用のコネクタ部です。別途販売される後付けの各種機能モジュールをここに取り付けることで様々な機能拡張をすることができます。拡張機能モジュールについては当方のWEBサイトから販売致します。

⑦リセットスイッチ

PICマイコンのMCLRピンと接続されており、J10のジャンパーソケットが、MCLR側に設定されている時、スイッチを押すとPICマイコンにハードウエアリセットがかかります。
MCLR機能を無効にしていたり、J10ジャンパーがI/O側に設定されている時には、リセットスイッチは動作しません。

⑧～⑫共通回路図

⑧ポート毎のピン状態表示LED(計36個)

PORTA～PORTE(以下RA～REと記載)の全ビットに接続されたLEDです。該当ピンがHレベルで点灯します。ピンの状態が視認でき、実験やプログラムの動作確認に大変役立ちます。

LEDのカソード側は、基板上のディップスイッチSW3を通してGNDに接続されています。SW3の各ポートのスイッチをON側にするとLEDが使用できるようになります。OFFで物理的に切断されます。
LEDを使用するためにSW3をONにしていると、PICマイコンのピンがHレベルの時LED側に電流が流れます。LEDを使用しない時はOFFに設定してご利用下さい。また別の回路とピンを共用する場合にはLEDがONになっていると電圧降下によって正しく電圧が周辺回路に印加されない場合がありますので、周辺回路を使う場合にはLEDをOFFにしてご利用頂くことをお奨め致します。

LEDはボード内の様々な回路に電流が流れると電圧下により点灯したり、うっすらと点灯することがあります。特にLCDがボードに取り付けられているとPORTBの各ビットはLCDの制御線と接続されており、LEDが点灯します。これは異常ではありません。LCDを取り外すとLEDは消灯します。LCDを使わない場合には取り外しておくことをお勧めします。

⑨各ビット個別プルアップ/プルダウン有効・無効設定スイッチ

各ポート、各ピン毎に設定できる「プルアップ、プルダウン、無効」設定ディップスイッチです。
スイッチを"PULL-UP"側に倒すと、該当ピンは抵抗器でVccにプルアップされます。スイッチを"PULL-DOWN"側に倒すと、該当ピンは抵抗器でGNDにプルダウンされます。
スイッチを中心位置にセットすると、そのピンはオープンとなります。

スイッチは
上位置: プルアップ
中位置: オープン
下位置: プルダウン
と3つに切り替えができます。

各ピンの⑪タクトスイッチを使う場合にはプルアップ、プルダウンの設定を適宜正しく設定する必要があります。

⑩各ピン取り出し用ヘッダーピン

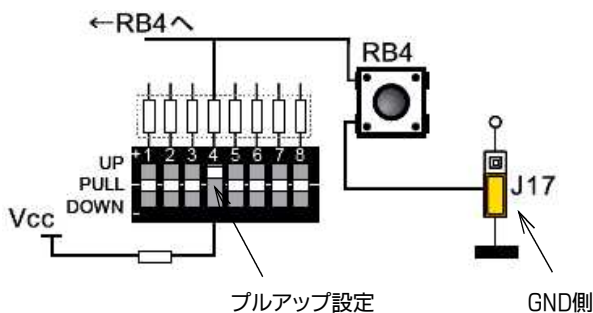
ピン毎にPICマイコンのピンを外部回路などと接続できるヘッダーピンです。同じピン配列のヘッダーピンが2つずつ実装されています。
シルク印刷の通りPICマイコンのピンにアクセスできますので、外部の回路と接続したり、オシロスコープ等の計測器と接続する場合などにお使い下さい。

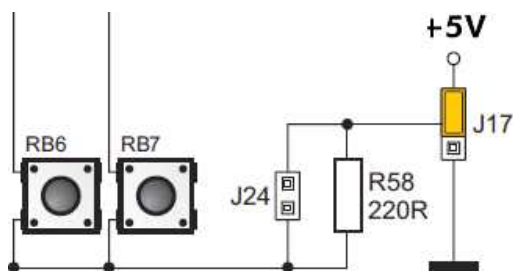
オシロスコープなど計測器を使用する場合には、PICD-700SX上に3つあるGNDピンを併用すると便利です。

⑪I/Oピン接続タクトスイッチ群(計36個)

J17のジャンパー設定によって、アクティブHigh又はアクティブLowが選択可能な36個のタクトスイッチです。
J17ジャンパーを"VCC"と刻印された側に設定すると、タクトスイッチを押した時、そのピンはVcc電位になります。
J17ジャンパーをGNDマークの刻印された側に設定すると、タクトスイッチを押した時、そのピンはGND電位になります。

各スイッチを使用する場合には、適切に各ピンのプルアップ又はプルダウンの設定と、上記J17の設定を組み合わせて使用し、スイッチの状態をアクティブHighか、アクティブLowかに設定して使用する必要があります。
※アクティブHighは、ボタンを押した時に該当のピンがHレベルになること、アクティブLowは該当のピンがLレベルになることです。

例えば、RB4のスイッチを使用する場合、ボタンを押していない時はHレベル、ボタンを押した時はLレベルになるように設定したい場合(アクティブLow設定)には、J17のジャンパーをGND側に設定し、続いてPORTBのプルアップ/ダウンディップスイッチの"RB4"を"PULL-UP"側に設定して、プルアップを有効に設定する必要があります。

上図の状態にしておくと、スイッチを押さない時はRB4はHレベル、スイッチを押すとLレベルになります。これをアクティブLowのスイッチと言います。

J24のジャンパーは、短絡時保護用抵抗を無効にするか、有効にするかを設定するものです。下図の通り、J24は220Ωの抵抗に並列に接続されており、ジャンパーソケットを外すと、タクトスイッチは220Ωの抵抗を介して+5V又はGNDと接続されます。

これによって例えば、PICマイコン側が意図せず出力設定(TRISレジスタにより)になっていた時にあるロジック状態になった際、誤ってタクトスイッチを押した時に、大きな電流が流れずに220Ωの抵抗を介して流すことができ、PICマイコンの破損を防止できます。
通常使用時は、J24のジャンパーは外しておくことを推奨します。

⑫I/Oピン取り出し用パターン

外部の回路などとPICマイコンのピンを接続する際に便利なパターンを用意しました。各ビットの信号線が取れます。
半田付けして2.54mmピッチの各種コネクタなどを取り付けることもできます。

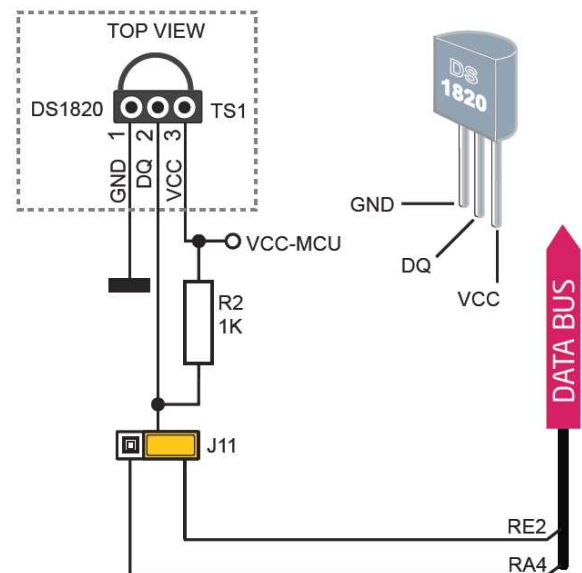
⑬⑮8ピン〜40ピンのPICマイコン取り付け用ソケット

⑬は40ピンPICマイコン用ソケットで、標準ではPIC18F45K22が装着されています。
この周辺には8ピン〜40ピンまでのDIP形状のPICマイコンが装着できるICソケットが配置されています。ソケットには複数のPICマイコンは装着できません。必ず1種類だけを装着して使用します。

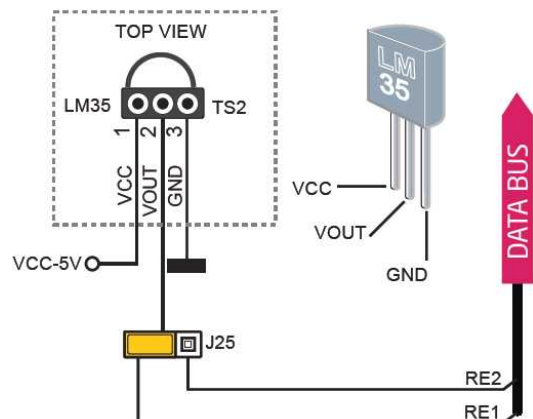
⑭外部発振子取り付け用ソケット

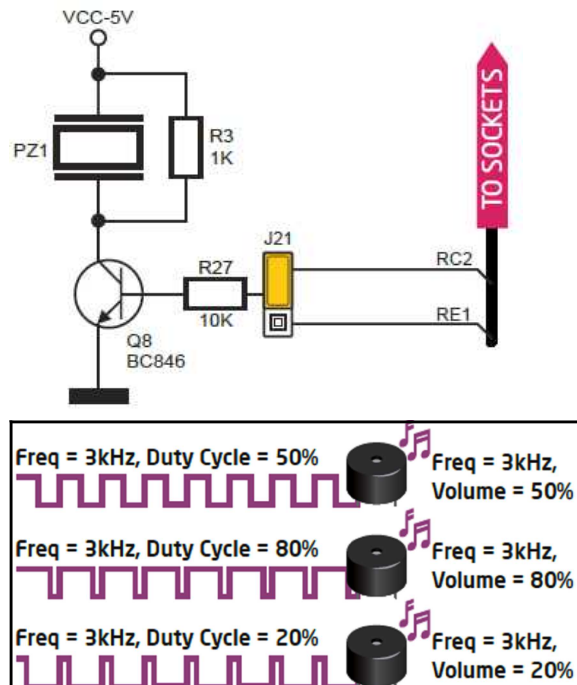
水晶発振子や、セラミック発振子等の外部発振子を取り付けるためのソケットです。標準では8MHzの水晶発振子が取り付けられています。
使い方の詳細は本書7ページ〜8ページをご覧ください。

⑯DS18S20、半導体温度センサIC装着用ソケット

オプションの半導体温度センサIC、DS18S20を装着するためのソケットです。
J11のジャンパーによって、DS1820のDQピンをRA4又はRE2に接続できます。なお、DQピンは、10kΩの抵抗でプルアップされています。使用しない場合には、ジャンパーソケットは外しておきます。

⑰LM35、アナログ温度センサIC装着ソケット

オプションのアナログ温度センサIC、LM35を取り付けるためのソケットです。LM35は、1℃当たり10.0mVという温度に比例した電圧を出力するICです。この電圧値をADコンバータで読むことで温度を計測できます。(−55℃〜+150℃まで対応しています)

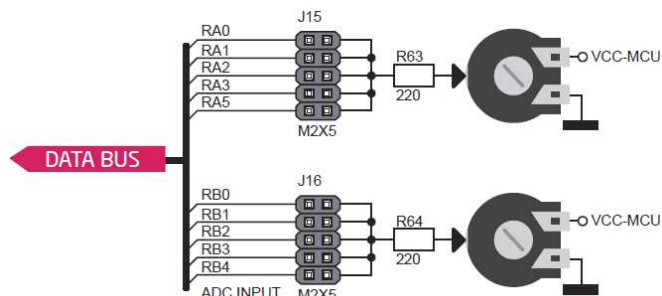
LM35のアナログ電圧出力は、J25によってPICマイコンのRE1又はRE2と接続することができます。使用しない場合にはジャンパーソケットは外しておいて下さい。

⑱圧電スピーカー

PWM信号を使って、音を出力する実験に使用できる圧電スピーカーです。NPNトランジスタ(BC846)によって増幅されて駆動されます。PICマイコンとはJ21の選択によってRC2又はRE1と接続して使用することができます。
PWMの周波数を変えることで音色が変化したり、デューティー比を変えることで音量が変化することなどが確認できます。

⑲⑳キャラクタLCD、GLCDコントラスト調節ボリューム

⑲は16×2行のキャラクタLCD用のコントラスト調整ボリュームです。回転させることでコントラストを調整できます。
⑳はGLCD用のコントラスト調整ボリュームです。適宜調整してご利用下さい。

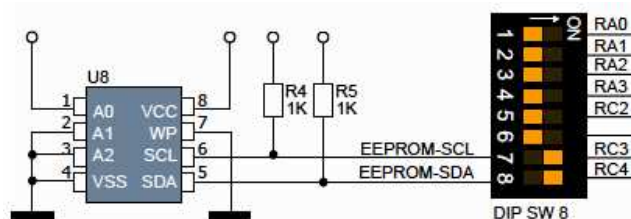
㉑A/Dコンバータ評価用ボリューム

A/Dコンバーター評価用のボリュームです。2つのボリュームを搭載しており2系統のA/Dコンバータの評価や実験ができます。
J15及びJ16のジャンパー設定により、指定したピンに0V～Vccまでの電圧を印加することができます。
印加できるピンはJ15側がRA0～RA5(RA4はありません)のいずれか、J16側がRB0～RB4のいずれかとなっています。

A/Dコンバーターを使用する場合には、アナログ入力として使用する該当ピンのプルアップ/ダウン設定をディップスイッチでオープンに設定してください。

㉒実験用I2Cインターフェイス EEPROM（24C08付属）

同期式シリアル通信のI2Cインターフェイスを採用したEEPROM、24C08が搭載された実験用EEROM回路です。
スレーブアドレスは001に設定されています。(A0～A2のデバイスセレクトビットはA0のみHで、他2つはLです。)

SCL、SDAの信号線は1kΩの抵抗器でプルアップされています。100KHz及び400KHzの速度に対応しています。

SCL、SDAのクロック線と信号線は、それぞれSW8のディップスイッチを介して、RC3及びRC4に接続されています。
このEEPROMを使用する場合には、ディップスイッチSW8の7番、8番スイッチをON位置に設定して物理的に配線を接続してご使用下さい。

付属の24C08は、EEPROMでは大変有名な種類で、色々なアプリケーションで使われています。EEPROMですので電源を切断しても内容は保持されます。400KHzまでのI2C通信に対応しています。

※EEPROMの使い方は別途データシートなどをご覧ください。

㉔4桁7セグメントLED(ダイナミック点灯方式)

コモンカソードタイプのLEDが4桁搭載されており、ダイナミック点灯方式で数値を表示させることができます。
7セグメントLEDは下図のようにセグメント毎にa～gまでのアルファベットが割り当てられており、各セグメントはPICマイコンのPORTDポートの各ビットに接続されています。

各桁のコモンピンは、トランジスターを通してRA0～RA3に接続されています。該当のビットをHレベルにすることで各桁の表示が有効になります。DIS0はRA0、DIS1はRA1･･･と対応しています。
RA0～RA3を、コモン駆動用のトランジスターと物理的に接続するか、接続しないかの設定は、SW4のディップスイッチにて設定できます。SW4のDIS0～DIS3と印刷されたスイッチをON側にすると、PICマイコンのピンとトランジスターが接続されます。
7セグメントとPICマイコンのピンは下のように接続されています。

7セグ側	PIC側の結線先
a	RD0
b	RD1
c	RD2
d	RD3
e	RD4
f	RD5
g	RD6
dp	RD7

数字を形成する場合にはPORTDの次のビットをHighレベルにします。

表示数字	Hレベルにするビット	PORTDの値（16進）
0	RD0、RD1、RD2、RD3、RD4、RD5	0x3F
1	RD1、RD2	0x06
2	RD0、RD1、RD2、RD3、RD4、RD6	0x5B
3	RD0、RD1、RD2、RD3、RD6	0x4F
4	RD1、RD2、RD5、RD6	0x66
5	RD0、RD2、RD3、RD5、RD6	0x6D
6	RD0、RD2、RD3、RD4、RD5、RD6	0x7D
7	RD0、RD1、RD2、RD5	0x27
8	RD0、RD1、RD2、RD3、RD4、RD5、RD6	0x7F
9	RD0、RD1、RD2、RD3、RD5、RD6	0x6F

※7セグメントLEDを使用する場合には、PORTAのプルアップ/プルダウン設定用のディップスイッチのRA0～RA3をOFF位置に設定してください。

※7セグメントLEDはPORTDを使いますので、PORTDのないピン数の少ないデバイスではご使用になれません。

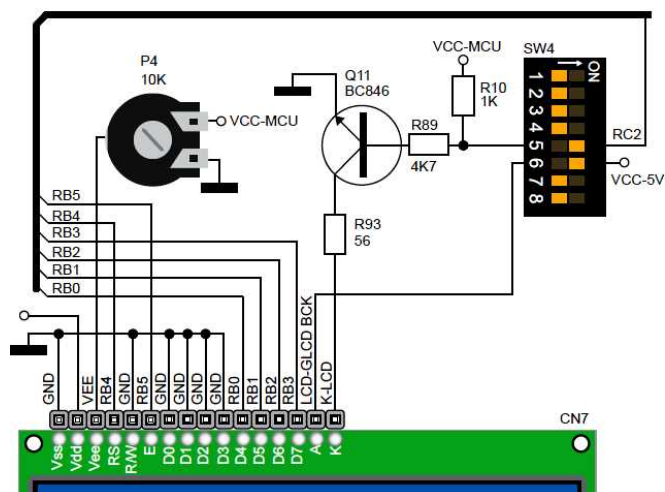
㉕2行16桁液晶ディスプレイユニット(4ビットモード)

2行16桁のLCDです。データ線にD7～D4の4ビットを使用する4ビットモードで使用できます。コントラスト調節は、P4ボリュームで行うことができます。

LCDのデータピンD4～D7はそれぞれPICマイコンのRB0～RB3に、EnableビットはRB5に、RSピンはRB4に接続されています。

LCDバックライトLEDは、アノード(+側)がディップスイッチSW4の6番スイッチにて、ON/OFFの設定が可能です。カソード(-側)はトランジスタを介してSW4の5番スイッチでPICマイコンのRC2と接続されます。これによって、RC2をPWM制御で駆動することでバックライトの照度を調整することができます。

常にバックライトを点灯させておく場合にはSW4の6番のみをON位置に設定しておきます。

バックライトの照度をPWM制御で調節したい場合にはSW4の5番と6番両方をON位置に設定し、PICマイコンのRC2からPWM信号を出力することで照度の調節ができます。

㉖グラフィックLCD(128x64ドット)接続ピン

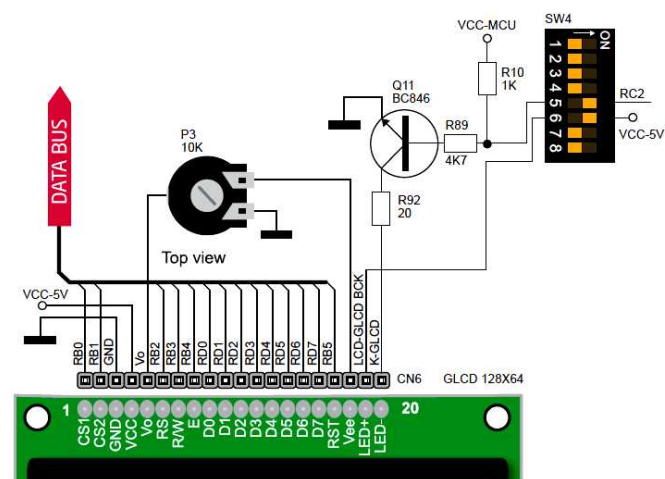
㉗4線式抵抗膜方式タッチパネル駆動用回路

128×64ドットのグラフィックLCDを接続するピンです。

LCDのコントラストは、P3のボリュームで調整します。

LCDバックライトLEDは、アノード(+側)がディップスイッチSW4の6番スイッチにて、ON/OFFの設定が可能です。カソード(-側)はトランジスタを介してSW4の5番スイッチでPICマイコンのRC2と接続されます。これによって、RC2をPWM制御で駆動することでバックライトの照度を調整することができます。

常にバックライトを点灯させておく場合にはSW4の6番のみをON位置に設定しておきます。

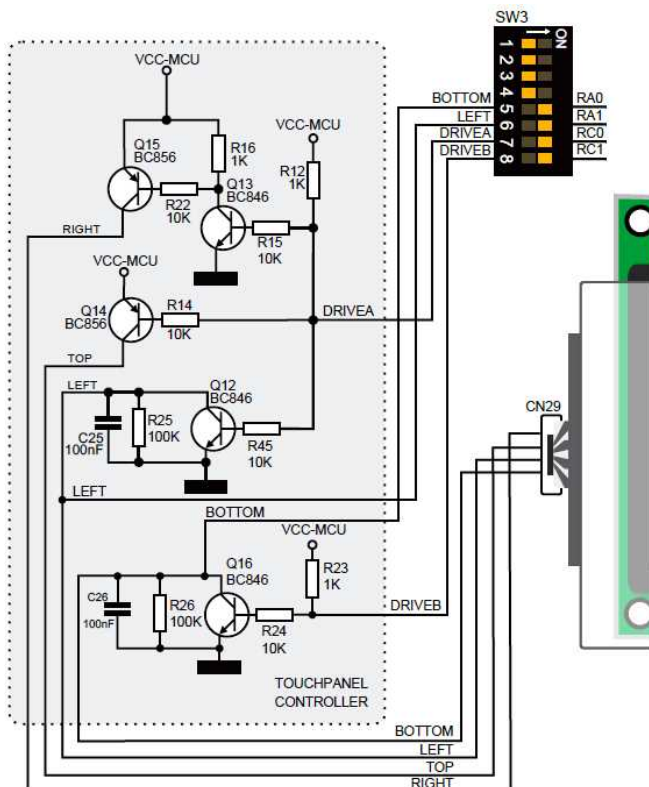
バックライトの照度をPWM制御で調節したい場合にはSW4の5番と6番両方をON位置に設定し、PICマイコンのRC2からPWM信号を出力することで照度の調節ができます。

4線式抵抗膜方式のタッチパネル搭載タイプのGLCDを装着する場合には、GLCDを装着する前に、タッチパネルから伸びるフラットケーブルをPICD-700SXのフラットケーブルコネクタに差し込んで装着してください。(詳しくは本書8ページをご参照下さい。)

フラットケーブルを装着した後、GLCDを取り付けます。タッチパネルの信号はタッチパネルコントローラ回路部に接続されます。

ドライブAとドライブBは、SW3の7番、8番スイッチを介してRC0とRC1へ接続されます。また、タッチ位置検出のBOTTOM信号はSW3の4番スイッチを介してRA0へ、LEFT信号はSW3の6番スイッチを介してRA1へ接続されます。

駆動回路のトランジスタのコレクターから伸びている配線は、タッチパネル用コネクタの4線に接続されています。

タッチパネルの使い方については、C言語でのサンプルプログラムがCD-ROMのサンプルプログラムフォルダに収録されておりますので、そちらをご参照ください。

なお、タッチパネル駆動回路とPICマイコンがSW3がONになって、接続された状態の場合、USBライターでのプログラムの書き込み実行時にエラーが発生することがあります。タッチパネルを使用しない時及び、USBライターでプログラムを書き込む場合には必ずSW3の該当スイッチをOFFにして、配線を物理的に切断した状態にしておいてください。

⑱外部信号取り出し用ヘッダピン

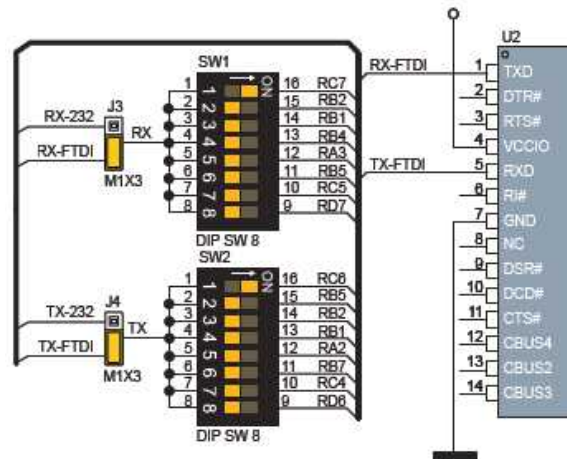
ポート毎の各ピンの信号を外部に取り出しやすいように実装されたヘッダーピンです。オシロスコープやテスターなどで各ピンの状態を観察したい時などに使用できます。

㉙USB-UART変換IC(FT232R)付き実験用UART

非同期式シリアル通信(UART)の信号をパソコンとやりとりする際に使用できるUSB-UART変換IC(FT232R)付きのUSBポートです。

UART信号は、パソコンと接続して通信させる時従来ではRS232C信号としてRS232Cレベル変換ICを介して通信させることが一般的でしたが、最近ではRS232Cポートを搭載していないパソコンが増えてきたため、UART信号をUSBポートからやりとりできるようにした回路を搭載しました。FT232RというUART信号をUSBポートでやりとりするための変換ICを搭載しています。
パソコン側には、仮想COMポートドライバーをインストールすることで、従来のRS232Cポートにアクセスするのと同じ間隔でUSBポートで接続されたUART通信デバイスにアクセスできます。

※ドライバのインストール方法については本書9ページをご覧ください。

UART信号線のTX線(送信データ)、RX線(受信データ)の選択は、ディップスイッチSW1とSW2によって設定します。SW1はRX線、SW2はTX線の選択スイッチです。
さらにUART信号線をRS232Cレベル変換IC経由でD-SUB9ピンで使用するか、USB-UART変換IC(FT232R)経由でUSBポートで使用するかをジャンパーJ3、J4にて設定します。
USB-UART変換ICを使ってUSBポートでUART信号の通信をしたい場合には、J3及びJ4のいずれも"USB UART"と書かれた下側にソケットを装着して使用します。

※パソコン側は仮想COMポートとしてUSBポートと通信ができます。

㉚RS232C通信ポート

レベル変換IC搭載のRS232Cポートです。
RX(受信データ)及びTX(送信データ)の信号線の接続先は、SW1及びSW2のディップスイッチによって、選択することができます。
さらにUART信号線をRS232Cレベル変換IC経由でD-SUB9ピンで使用するか、USB-UART変換IC(FT232R)経由でUSBポートで使用するかをジャンパーJ3、J4にて設定します。
D-SUB9ピンのRS232Cポートを使う場合には、J3及びJ4のいずれも"RS232"と書かれた上側にソケットを装着して使用します。

RS232C通信の実験をする場合には、パソコンとD-SUB9ピンの全結線ストレートケーブルをご使用下さい。

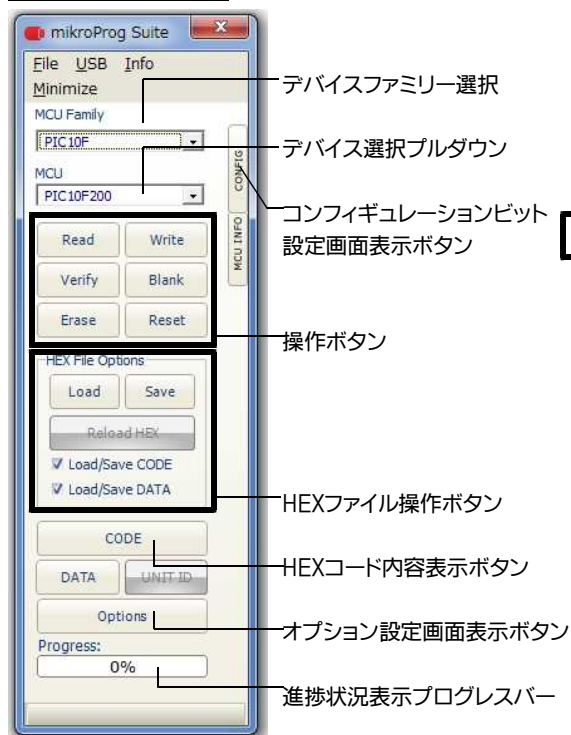
書き込みソフトウエア(mikroProg Suite)の使用方法

付属のPICプログラマー、mikroProg Suiteの使い方を紹介します。mikroProg Suiteは、mikroCやMPLAB等で作成したHEXファイルをPICマイコンへ書き込む際に使用するソフトウエアです。

■mikroProg Suiteの起動

PICプログラマーは、Windowsのスタートメニュー又はデスクトップに作成されたショートカットから起動できます。

スタートメニューから起動する場合には、"スタート"→"プログラム"→"Mikroelektronika"→"mikroProg Suite For PIC"→"mikroProg Suite For PIC"の順でクリックして起動します。

■アプリケーション概要

初期画面　　　"CONFIG"ボタンを押した時に右側に表示される画面

■操作の順番

次の順番でHEXファイルを書き込みます。

1 デバイス設定プルダウンより、書き込むデバイスを選択します。
最初に"デバイスファミリー選択"プルダウンからデバイスファミリーを選択します。PIC10/12/16/18と選択できます。

2 続いて"デバイス選択プルダウン"より書き込みを行うデバイスを選びます。

3 書き込むHEXファイルを読み込みます。
HEXファイル操作ボタンの"Load"ボタンを押します。
ダイアログが表示されますので、ファイルを選択します。

→ファイルが読み込まれるとステータスバーに、ファイル名が表示されます。

※同じファイルを再度ロードする場合には、いちいち上記のようにファイルを指定しなくても、"Reload HEX"ボタンを押すことで再読込ができます。

※ファイル名や、ファイルの配置してあるディレクトリ名に日本語や2バイト文字が含まれている場合、正しくHEXファイルの読み込みができません。ファイル名及びディレクトリ名は必ず半角英数字になるようご注意ください。

4 一般的にはこれで書き込み準備は完了ですが、必要に応じてコンフィギュレーションレジスタの設定を確認します。
コンフィギュレーションレジスタは、プログラムを書き込む時にだけしか設定できないPICマイコン全体の動作や基本的な設定などを行う特殊なレジスタです。
コンフィギュレーションビットの設定が間違っているとプログラムや回路が正しくてもプログラムは動作しません。通常、コンフィギュレーションビットの設定内容は、プログラム内に埋め込まれているため、HEXファイルを読み込んだ時点で自動的に設定されます。
しかし、設定が正しくないとプログラムの動作に問題が発生することがありますので、書き込み前に確認されることをお奨めします。

"コンフィギュレーションビット設定画面表示ボタン"をクリックします。右側に設定ウインドウが展開します。

コンフィギュレーションレジスタの内容はデバイスに搭載される機能により変わるため、設定項目もデバイスにより様々です。ここでは基本的な設定項目についてのみ説明します。

■Oscillator
LP･･･低電力水晶 200KHz以下
XT･･･水晶発振子 4MHz以下
HS･･･高周波水晶、セラミック発振子 4MHz～20MHz
RC･･･RC発振(5KΩと20pFの組み合わせで約1MHz)
EC･･･外部発振子よりTTLレベルのクロック注入
H4･･･HS+PLL内部PLLにてクロックアップ(10MHz発振子接続)
INTRC(IN)･･内部発振,外部に発振子を取り付ける必要なし

■Watchdog Timer Enable
ウォッチドッグタイマのOn/Offを設定します。通常はDisabledに設定します。

■PowerUp Timer
デバイスの電源投入時には、クロックが不安定で動作が不安定になります。この機能を有効にすると、電源投入時一定時間リセットをかけ続けることで、電源が安定した後にクロック動作を開始する機能です。通常はEnabledに設定します。

■Brown Out Detect
一時的な電圧降下の時にハードウエアをリセットする機能です。
PowerUp Timerが有効の時のみ使用できます。

■Low Voltage Programming
低電圧書き込み機能を使用するかどうかを設定します。使用しない場合には必ず "Disable" に設定してください。

■MCLR function
デバイスのMCLR機能(ハードウエアリセット)を有効にするか無効にするかを設定する項目です。Enabledに設定すると、MCLRピンをGNDに接続するとPICはハードウエアリセットされます。
Disabledに設定すると、MCLRピンは入力ピンとして利用可能になります。

[注意]水晶発振子等をお使いで4MHzより高い周波数でお使いの場合には必ずOscillatorの種類をHSに設定してください。またセラミック発振子(レゾネータ)をお使いの場合には周波数に関係なくHSに設定してください。設定が正しくない場合PICマイコンは動作しません。

※外部発振子か内蔵発振子を使用するかにより、J13及びJ14の設定が異なりますので、必ず使用前に設定をご確認下さい。

※内蔵発振子(INTOSC)を使用する場合には、デバイスの種類によってはプログラム内でOSCCONレジスタを設定しなければならない場合があります。詳しくは、使用するデバイスのデータシートをご覧下さい。

5 必要に応じてコード保護機能も設定します。
コードプロテクションを有効にすると、一度書き込んだHEXデータは読み出せなくなります。コードを不正にコピーされることを防止できます。通常は、"None"及び"Write protection Off"に設定します。

6 設定が完了、確認したら書き込みを行います。
PICD-700SX本体の"USB LINK"の黄LEDが点灯していることを確認してから操作ボタンの"Write"ボタンをクリックします。
書き込みを開始します。
書き込み動作中は、進捗表示プログレスバーに進捗状況が表示されます。完了するまで待ちます。

7 書き込みが完了すると、プログラムがすぐにボード上で動作を開始します。

■その他の機能
・Read機能
→PICマイコンからデータを読み込みます。
読み込んだデータは、"Code"ボタンで閲覧できます。またメニューバーの"File"→"Save HEX"で保存できます。

・Verify機能
→現在読み込まれているHEXファイルの内容と、PICに書き込まれているプログラムの内容が一致しているか検証します。

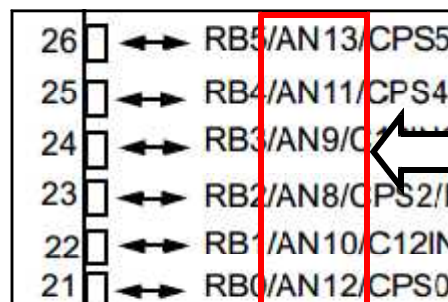
・Blankチェック機能
→現在装着されているデバイスのプログラムメモリがブランク(空)かチェックします。

・Erase機能
→現在装着されているデバイスのプログラムメモリの内容を消去します。

よくあるトラブル事例

以下にPICD-700SXを使用する上での注意事項をまとめました。

■RA4のLEDが点灯しない
RA4は、ほとんどのPICマイコンではI/Oピンのドライバーがオープンドレイン方式です。そのため、その他のピンと同様、TTLのように扱おうとしても動作しません。すなわち、このピンはHレベルやLレベルを出力するのではなく、Vssに接続されたFETのオン/オフを出力しています。よってこのピンからHレベルやLレベルをそのまま得ることはできません。

■PORTAやPORTBがプログラム通りに動かない
PORTAやPORTBなどはデバイスによっては、ADコンバータらアナログ電圧入力ポートにアサインされておりレジスタを設定しないとデジタルI/Oピンとして使用できません。

使うPICマイコンのデータシートで、ピンアサインの図を見たとき、"AN○"と書かれたピンは、電源投入時はすべてアナログピンになっています。これらのピンをデジタルI/Oピンとして使う場合には必ずプログラム内でレジスタの設定が必要です。
設定はデバイスによって異なります。例えばPICD-700SXに標準で付属のPIC16F1939の場合には、
ANSELA = 0;
ANSELB = 0;
などANSEL○レジスタを0にします。PIC16F887など比較的古いデバイスでは、「ADCON1 = 7;」などの設定になります。この辺りの設定値についてはデータシートに書かれていますので、データシートのADコンバータについて書かれたページを参照してください。

■書き込んだプログラムが動かない

原因は様々ですが、まずは下記の点を再度ご確認下さい。

①Oscillatorの設定は正しいですか?

→5MHz以上の水晶発振子の場合及び周波数に関係なくセラミック発振子の場合にはOscillatorの種類は、HSに設定してください。

②J13及びJ14の設定及びOSC1、OSC2は正しいですか?

③MCLRピンの設定及びJ7の設定は正しいですか?

④J8及びJ9の設定は正しいですか?

■該当ピンをHighにしたのにVcc電圧(+5V)が出力されない

Highレベルに設定したピンがLEDと接続されていませんか?
LEDと接続されていると、LED側に電流が流れるため電圧降下が発生して、ピンの電圧は下がります。SW6でLEDと切り離してお試しください。

■スイッチのアクティブLow、Highが設定通りにピンに現れない

ADコンバータ用のジャンパー、J15・J16がジャンパーソケットでADコンバータ用のボリュームと接続されていませんか?
その他、ジャンパーソケットが意図しない設定になっていないか確認ください。

■急に書き込みができなくなった、エラーが表示されるようになった

①パソコンを再起動してから、再度お試しください。

②PICD-700SXのデバイスドライバーを入れ直してください。

③何か外部回路を接続している場合にはすべて取り外しておためしください。

■PICプログラマーでHEXファイルが正しく読み込めない

PICプログラマーでは、長い名前のファイルや階層が深いディレクトリにあるファイルを読み込めない場合があります。
HEXファイルのファイル名は短く、ファイルの保存場所はなるべくルートディレクトリに近い場所に保存してください。またディレクトリ名やファイル名に日本語などの2バイト文字を使用している場合には文字コードの関係上正しく読み込みができません。ディレクトリ名やファイル名には半角英数を使用するようにしてください。

■時間遅延関数(delay_msなど)が正しく動作しない

時間遅延系の関数が、プログラムした通りの時間遅延しない場合には、動作クロックとコンパイラ上の設定クロックが一致していないことが考えられます。例えば、実際にPICマイコンに供給されるクロックが8MHzだった時に、コンパイラ側で16MHzのクロック周波数と設定してしまった場合、プログラムの実行速度は1/2になります。例えば100ミリ秒の遅延のプログラムを記述すると実際には200ミリ秒の遅延になります。PICマイコンに供給しているクロック周波数とコンパイラ側で指定している周波数が一致していることを確認下さい。
また、内蔵発振子を使う場合やPLLにて逓倍するような場合には、オシロスコープ等で実際の動作クロックをよく確かめながら作業を行って下さい。

「プログラムが動く」ことと「書き込みが成功した」ということは別物!?

よくあるご質問に「プログラムを書き込んだが、プログラムが動かない。書き込みができない。」というものがあります。しかし、「プログラムが動く」ということと、「HEXファイルの書き込みができている」ということとはイコールではありません。
端的に言えば、プログラムに誤りがあれば「書き込みが成功」していてもプログラムは動きません。

HEXファイルが正しく書き込めたかどうかの確認は、「プログラムが動作するかどうか」ではなく、ベリファイ機能を使って行います。
ベリファイとは、照合確認のことです。ベリファイを実行すると、マイコンライターは、一度PICマイコンに書き込んだファイルを読み出し、もとのHEXファイルと照合します。1ビットでも元のHEXファイルと書き込んだ内容に違いがあればベリファイでエラーが表示されます。ベリファイでエラーが表示されなければ「少なくともHEXファイルは全て正しくPICマイコンに書き込めている」ということになります。

ベリファイでエラーが表示されず、プログラムが動かない、という場合にはプログラムに問題があるということになります。

なおベリファイは、デフォルト設定では書き込み時(Writeボタンを押した時)に自動的に実行されます。よって書き込みを行って、エラーが出なければベリファイも成功しているということになります。
手動でベリファイをしたい場合には、書き込みソフトウエアの"Verify"ボタンを押してください。

エラーが表示されなければ成功です。(エラーがあった時のみメッセージが表示されます。成功の場合には何も表示されません。)

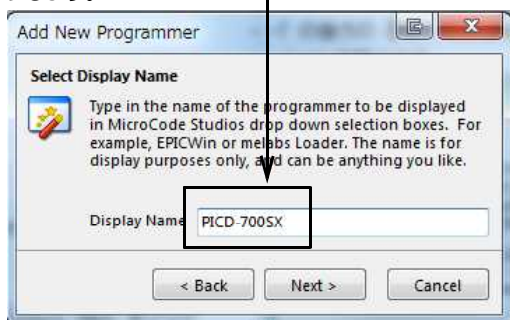
PBP3の開発環境、MicroCode Studio(Plus)から連動させてmikroProg Suiteを動作させる設定方法

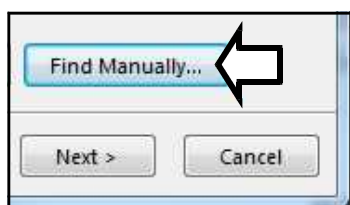
当方販売中のPIC用BASICコンパイラ、PICBASIC PRO CompilerVer.3(以下PBP3と記載)用の統合開発環境、MicroCode StudioX(Plus版も含む)から、PICD-700SXの書き込みソフトウエア、mikroProg Suiteを連動させて動作させるための設定です。PBPをお使いの方は、ここで設定を行うことによってMCS上から連携動作をさせることができるようになります。

1 MCSを起動します。メニューバーの"View"→"Compile and Program Options..."をクリックして開きます。

2 "Programmer Options"という部分が下にありますので、"Add New Programmer"ボタンをクリックします。

3 "Create a custom programmer entry"にチェックを入れて、"Next"をクリックします。

4 "Display Name"の欄に "PICD-700SX" と入力して"Next"をクリックします。

5 "Programmer Filename"の欄に "mikroProg Suite for PIC.exe" と入力して"Next"をクリックします。

6 PICD-700SX用の書き込みソフトウエアがインストールされているディレクトリを手動で設定します。
"Find Manually"ボタンをクリックします。

"Select Folder"ダイアログが表示されますのでPICD-700SXがインストールされているディレクトリを指定します。デフォルト設定でインストールした場合には通常下記のディレクトリになっています。

C:¥Users¥Public¥Documents¥Mikroelektronika¥
mikroProg Suite For PIC

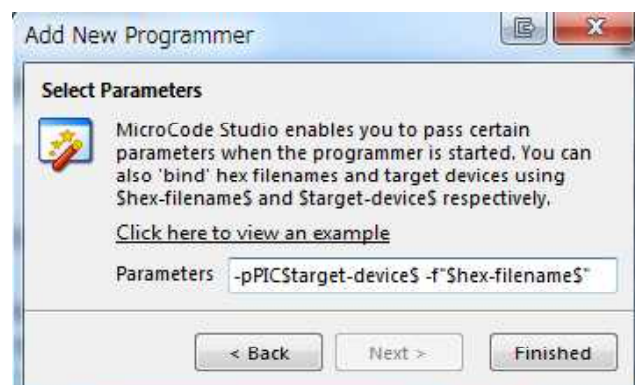
※インストールする時にインストール先ディレクトリをデフォルト設定以外から変更している場合には、適宜インストールされているディレクトリを設定してください。

"OK"ボタンと"Next"を押して続行します。

7 最後に"Select Parameters"ボックスが表示されます。
"Parameters"の欄に、下記の文字列を記述します。文字列には大文字小文字が区別されます。またスペースの位置など間違えないように注意して入力してください。

-pPIC$target-device$ -f"$hex-filename$"

↑
半角スペース1つ

-fのハイフンの前には半角スペースを1つ入力します。

-f の後ろの $hex-filenames$ の部分は必ず " ダブルクォーテーションで囲みます。

8 入力が完了したら"Finish"をクリックして終了します。
これで、MCSX(Plus)から"Compile Program"ボタンを押した時、コンパイルと同時にPICD-700SXの書き込みソフトウエア、mikroProg Suiteが自動的に起動します。起動したら"Write"ボタンを押して、書き込みを実行します。

主な仕様

電源電圧:	USBバスパワー給電時 DC5V ACアダプタ給電時 DC9V~DC12V
給電方法:	USBバスパワー又はACアダプタ
USB規格:	Ver,2.0対応
対応OS:	WindowsVISTA/7/8.1 32ビット、64ビット対応
対応デバイス:	PICマイコン DIPパッケージ
生産国:	セルビア

サポート情報

PICD-700SXのサポートを行っております。以下のいずれかの方法でご質問をお寄せください。

■FAX番号　　03-3700-3548
■電子メール　support@microtechnica.net

ご質問時には、できるだけ詳しくその内容をお書きください。例えば使用しているPICマイコンの型式やパソコンのOS、エラーメッセージが表示される場合には、その内容などをお知らせください。
なお、他社製品に関することや自作回路に関するご質問にはお答え致しかねますのであらかじめご了承ください。

ソフトウエアはアップグレードされることがあります。アップグレードされると対応デバイスが増えたりバグが修正されたりします。
アップグレードの情報などは当方のwebページにてお知らせ致しますので、定期的にご確認ください。

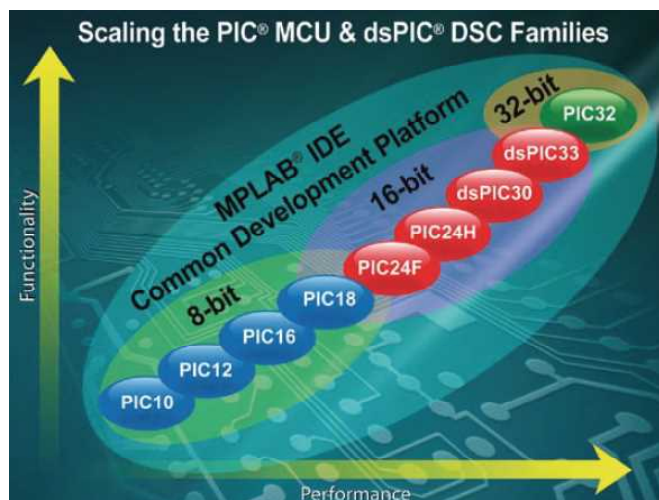
マイクロテクニカ
〒158-0094　東京都世田谷区玉川1-3-10
TEL: 03-3700-3535　FAX: 03-3700-3548

PICマイコンの概要

PICマイコンは、数あるマイコンのうちの1つでアメリカのマイクロチップテクノロジー社が開発しました。"Peripheral Interface Controller"の略でマイコンの中でも特にワンチップマイコンと呼ばれています。

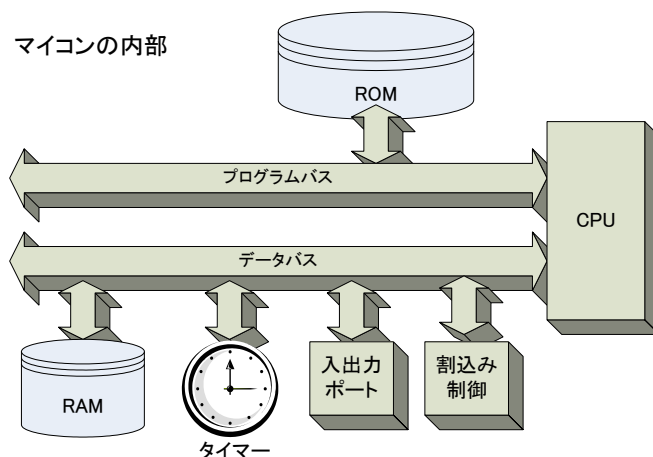
これはその名の通りワンチップ、すなわち1つのIC本体の中にほぼすべての機能が詰め込まれていることをいいます。PICマイコンはマイコンの動作に必要な機能(ROMやRAM、I/Oポートドライバー等)をワンチップ内にすべて入れてあり、ユーザーインターフェイスであるI/Oポートだけが外部端子に並んでいるのでプログラムが書き込まれるとすぐに動作を開始するという特徴があります。その使いやすさと手軽さ、入手製のよさなどから、世界中で使用されています。

PICマイコンには多くの種類があり、今では100種類以上が生産されています。ピン数も8ピンから、80ピン以上もある高機能版まで多種多様です。ピン数の違いの他にもそれぞれ機能の違いがあり、プログラムメモリの容量が違ったりA/Dコンバータを搭載していたり、USB通信機能を搭載していたりと・・・その選択肢の多さもPICマイコンの特徴の1つとなっています。PICマイコンのアプリケーションを作る際には、まずデバイスの選定からはじめるのが基本です。ただ、種類は多いといってもよく使用される"定番デバイス"というのがあります。その定番デバイスを知っておけば、より簡単にPICマイコンを使うことができます。

PICマイコンには8ビット、16ビット、32ビットのデバイスがあります。さらに16ビットデバイスにはデジタル信号処理用のプロセッサを搭載したdsPICシリーズがあります。

PICD-700SXは、このうち8ビットアーキテクチャに該当するデバイスに対応しています。このように見ると8ビットは下位のデバイスであるように見えますが決してそのようなことはありません。現在でも8ビットファミリのデバイスは精力的に開発が続けられており、世界で最も使用されているPICマイコンは8ビットタイプです。

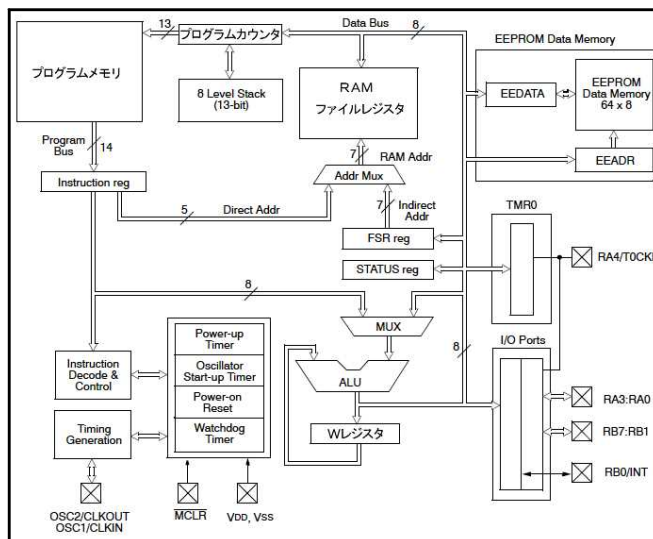
PICマイコンの構造

PICマイコンの基本的な構造すなわちアーキテクチャは他の一般的なコンピュータのノイマン型アーキテクチャと異なり"ハーバードアーキテクチャ"という構成を採用しています。

ハーバードアーキテクチャの特徴はプログラムを格納するメモリ(プログラムメモリ)とデータを格納するメモリ(RAM=ファイルレジスタやデータメモリと呼びます)が別々になっており,それぞれのメモリから読み出したり書き込んだりするためのデータの通信路であるデータバスが別々になっているという点にあります。

このアーキテクチャのメリットとしてはデータバスを別々にすることでプログラムの命令が何ビットの長さになっても、1回でメモリから読み出すことができるため簡単な構成でかつ高速に命令を実行できる点にあります。他のコンピュータ(ノイマン構造)では、データ長が8ビット単位であることが一般的なため命令も8ビットの倍数とせざるを得なくなり1つの命令が複数のバイト構成となるためにメモリからの呼び出し1回だけでは命令全体が読み出すことができなかったり、データとの切替えが都度必要だったと効率が悪くなります。こうなると当然複雑な内部構成となってしまい処理も遅くなってしまいます。PICマイコンではハーバードアーキテクチャの採用により12ビット・14ビット・16ビットの3種類のモデルを作り効率のよい命令実行を行っています。

プログラム大まかな実行の流れとしては命令がROM(プログラムメモリ)から呼び出されCPU部分で実行されます。その命令の内容によりデータがRAM(データメモリ)から呼び出されCPUで演算されて結果がまたRAMに格納されたりI/Oポートにデータが出力されたりします。

実際のマイコンの内部ブロック図を見てみましょう。PIC16F84Aデバイスの内部ブロック図を次に示します。

※PIC16F84AはPICマイコンの中でもシンプルな18ピンデバイスです

●プログラムメモリ(ROM)

プログラムそのものの命令を記憶する場所です。プログラムメモリと呼ばれたり単にROMと呼ばれたりします。ここにプログラムを書き込むためには専用のPICマイコンライターが必要となります。

●プログラムカウンタ

プログラムの実行順序を制御するカウンタ。このカウンタの内容が指しているアドレスのプログラムメモリにある命令が次に実行される命令となります。

●ファイルレジスタ(データメモリ)
ファイルレジスタはデータメモリと呼ばれたり単にRAMと呼ばれたりと、名称が混同されていますが,厳密には違いがあります。
ファイルレジスタは2つの領域に別れていて1つは汎用のデータ格納メモリエリアでRAMと呼ばれ、プログラム内で使う変数領域として使用します。もう一つは特別なレジスタ領域で一般的に"SFR"(Special Function Register)と呼ばれ、コンピュータ内の動作を決める色々な条件を設定したり、動作状態を知ることができます。
よって、正確にはファイルレジスタはRAMとSFRから構成されていると理解しましょう。

●Instruction Decode and Control
プログラムカウンタが指すプログラムメモリ内の命令が読み出されここで命令種類の解読と処理がなされます。その結果,各種の制御やALU内で演算をしたりすることで命令が実行されます。

●MUX(切り替え部)とALU(算術演算部)
各命令の指示に従って各種レジスタやWレジスタの内容との演算がなされる場所で、いわゆるコンピュータの中の計算器です。演算結果は再度Wレジスタやレジスタに格納されたりI/Oポートやタイマ、プログラムカウンタなどに格納されます。

●Wレジスタ(W reg)
演算をするときに一時保管用に使うレジスタで演算の時の中心となって働きます。

●入出力ポート(I/Oポート)
内部のデータを外部に出すためのポートでこれがPICマイコンピンに直結されています。
そのためVccが5Vの時はプログラムがポートに"1"のデータを出力すると対応するピンが"High"レベルの状態になり、"0"を出力すると"Low"の状態となります。

PICマイコンの種類

PICマイコンには様々な種類があり、選択の幅が広いマイクロコントローラーです。ピン数も8ピンから80ピンのものまであり、用途に応じて使い分けることができます。

PIC10FシリーズはPICの中でも最も小さいマイコンです。DIP形状のものは8ピンからあります。PIC12Fシリーズは、8ピンながら高機能でADコンバータを搭載したモデルもあります。

PIC16Fシリーズは、PICのスタンダードモデルです。14ピン～40ピンまで数多くのデバイスが流通しています。以前はPIC16F84などが主流でしたが最近では高機能なPIC16F88や18ピンの代表格となっています。
PIC16シリーズはPIC18シリーズの下位レンジとして位置づけられていましたが、最近になってPIC16F19xxシリーズが登場すると、格段に機能が強化され動作周波数もこれまでの最高20MHzから32MHzへと高速化が進展しました。さらに内蔵発振子で32MHzが実現できることや、デバイス単価が安いこともあり、今後PICマイコンの分野でPIC16F19xxシリーズは中心的な存在となっていことが考えられます。

PIC18Fシリーズは、PIC16Fシリーズを高機能化したもので様々な機能が付加されている他、最大の特徴はバンクの概念がなくなったことです。バンクの概念は高級言語を使用している分にはそれほど問題にはなりませんが、アセンブラを使用している場合には、かなりプログラムの仕組みが簡単になりました。

USB2.0機能搭載のデバイスや、ROM容量の大きなデバイス、ピン数が多い80ピンのデバイスなど、PIC16Fシリーズでは事足りない場合の選択肢としてPIC18Fシリーズがあります。

これまでのPIC10F、12F、16F、18Fシリーズはいずれも8ビットMPUです。8ビットマイコンでは、世界シェアNo.1であり市場の18%程度をPICが占めています。
PICも高機能化、高速化の波にのり最近では、16ビットマイコンのデバイスも登場しています。PIC24シリーズは16ビットPICマイコンの代表格です。dsPIC30F、33Fシリーズは16ビットマイコンにDSP機能を追加した特殊なマイコンで、デジタル信号処理を得意としています。
また、最近では32ビットのPICも登場し、PIC32として注目を集めいてます。但し、32ビットマイコンの世界ではすでに多くのメーカーから高機能なCPUが発売されており、PICの特徴を活かせるかは今後の展開によるところです。

マイクロコントローラーの世界では「大は小を兼ねません」。適材適所である必要があります。開発するシステムの規模に応じて、最適なデバイスを選択することで、性能だけではなく消費電力や価格、デバイスサイズなど、色々な問題を最適化することができるためです。PICは豊富なラインナップにより、様々なニーズに応えられるデバイスとして世界的に有名になりました。

PICマイコンの動作概要

プログラムは常に0番地からスタートします。電源投入時やハードウエアリセットがかけられた状態がこれに相当します。
プログラムの進行はプログラムカウンタによって制御されています。プログラムカウンタの指し示す値が実行される番地を示しています。リセット時プログラムカウンタは0になり、0番地の命令より順番に実行されていきます。1つの命令が読み出されると演算ユニットALUにて演算や処理が行われます。必要に応じてデータメモリからデータを読み出したり書き込んだりも行います。
演算ユニットはプログラム側から見るとWレジスタの形で存在している点に注意が必要です。I/Oポートは,データメモリ上に割り当てられていてメモリ操作によってアクセスできます。この構造のことを特に"メモリマップドIO"と読んでいます。すなわち,メモリに0や1の値を書き込むことで直接I/OポートがLレベルになったりHレベルになったりします。

PICマイコンはRISC(Reduced Instruction Set Computer)タイプのマイコンなので1命令1クロックで動作します。クロックとは外部から入力されたある一定の周期の矩形波パルスで通常は水晶発振子やセラミック発振子といった発振子を用いてクロックを生成します。デバイスによって20MHz～40MHz程度のクロックを扱うことができます。また、外部クロックを内部の逓倍回路(PLL)によって逓倍して周波数を高くするデバイスもあります。その他、発振子を内蔵したタイプもあり、外部に発振子を取り付けずに動作可能なデバイスも増えてきました。

プログラムメモリから命令を取り出すことを"命令をフェッチする"といいます。PICマイコンでは1クロック毎に命令はフェッチされています。フェッチされた命令はインストラクションデーコーダによって解析されます。なお,PICマイコンは内部で発振子より与えられたクロック周波数を1/4にして命令を実行しています。すなわち,4MHzのクロックを入力している場合には内部動作は1MHz動作になっています。

では実際の動作はプログラムを作りながら学習しましょう。

開発言語

PICマイコンの開発言語としては、アセンブラ言語・BASIC言語・C言語などがあります。アセンブラ言語は、フリーで使用できますが記述内容が直感的でなく習得には時間がかかります。そのかわり、マイコンの内部処理を1つずつ記述していくため、マイコンの仕組みや処理の仕組みが理解できます。

日常作業では、高級言語(C言語やBASIC言語など)を使用する技術者でもアセンブラ言語を理解しておくと何かと役に立ちます。

BASIC言語を使用するには、BASICコンパイラーが必要です。当方では、PICマイコン用に開発された、高機能なBASICコンパイラー、PicBasic Pro Compilerを販売しております。直感的に記述できる書式、豊富な組込命令を搭載しており、PICマイコンの機能をフルに使用できます。高度な機能を簡単な記述で実現できます。

C言語を使用するには、Cコンパイラーが必要です。本製品のCD-ROMには2Kワード限定版の体験版Cコンパイラー、mikroC PRO for PICを収録しており、すぐに体験することができます。mikroCは、ANSI規格に準拠したCコンパイラーで、豊富な組み込み関数を備えており、高度な機能を簡単な記述で実現できます。

本書では、このmikroCを用いてPICの世界を簡単に体験できるよう、以降にチュートリアルを収録しました。是非一度お試し下さい。

PICD-700SXのチュートリアル　～C言語編～

■C言語とは

C言語は、ANSIという団体によって規格化された高級言語です。高級言語とはアセンブラ言語のように機械語に近いレベルの開発言語ではなく、より人間がわかりやすい言語体系の開発言語を指します。C言語を導入することでより簡単に、合理的にPICマイコンの開発ができるようになります。

C言語では様々なデータ型を扱えるほか、データを指し示すポインタや配列、関数などが使用でき、より高度なプログラムを記述することができます。

本製品に付属のmikroCはANSI規格に準拠したPICマイコン用のC言語です。PIC10F/12F/16F/18Fシリーズの8ビットPICに対応しています。(この他16ビットPIC対応版もあります。)体験版のため、開発できるプログラムのコードサイズが最大2Kワードに限定されていること以外、すべての機能が使用できます。mikroCが気に入れば、優待価格にてサイズ制限のないフル機能版を入手することができます。

mikroCは、一般的なC言語としての機能の他にPICマイコンの機能をフルに使用できるよう様々な組み込み関数を搭載しています。組み込み関数を使用することで、LCD制御やADコンバーターの操作、UARTなど様々な機能を簡単に実現できます。

また、ICD機能(インサーキットデバッグ)が搭載されており、本PICD-700SXと組み合わせて使用すると、実機でプログラムを動作させながらプログラムの挙動を観察したり、1行ずつプログラムを実行させるステップ実行ができるなど、かなりの高機能を実現しています。一般的にはICD機能を行うには別途システムが必要でしたが、本PICD-700SXとmikroCの組み合わせですぐに実現できます。

C言語チュートリアル　～はじめに～

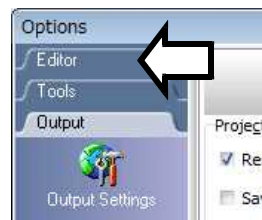
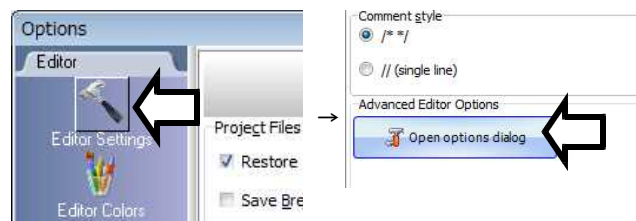
では、早速付属のmikroCを使ってC言語でプログラムを作って実際にPICD-700SXのボード上で動作させてみましょう。

本マニュアルではC言語の基本から解説することはできませんので、C言語の基本については専門書籍をご参照ください。

■開発環境の設定

では最初にmikroC Cコンパイラーの開発環境を設定しましょう。

1 mikroCを起動します。
"スタート"ボタン→"プログラム"→"Mikroelektronika"→"mikroC PRO for PIC"→"mikroC PRO for PIC"の順でクリックします。

2 起動すると、開発画面が表示されます。
初期設定のままでもかまいませんが、エディタウインドウの文字を大きくしたい場合等は、最初に環境を設定します。
メニューバーの"Tools"→"Options"をクリックします。画面左側のタブから"Editor"をクリックします。

3 メニューが展開しますので、その中から"Editor Settings"をクリックします。その中にある"Open Option dialog"ボタンをクリックします。

4 "Editor Options"ダイアログが表示されますので、画面右下の"Editor Font"部分にある"Font"ボタンをクリックすると、フォントを選択できるダイアログが表示されますので、サイズを調節してください。

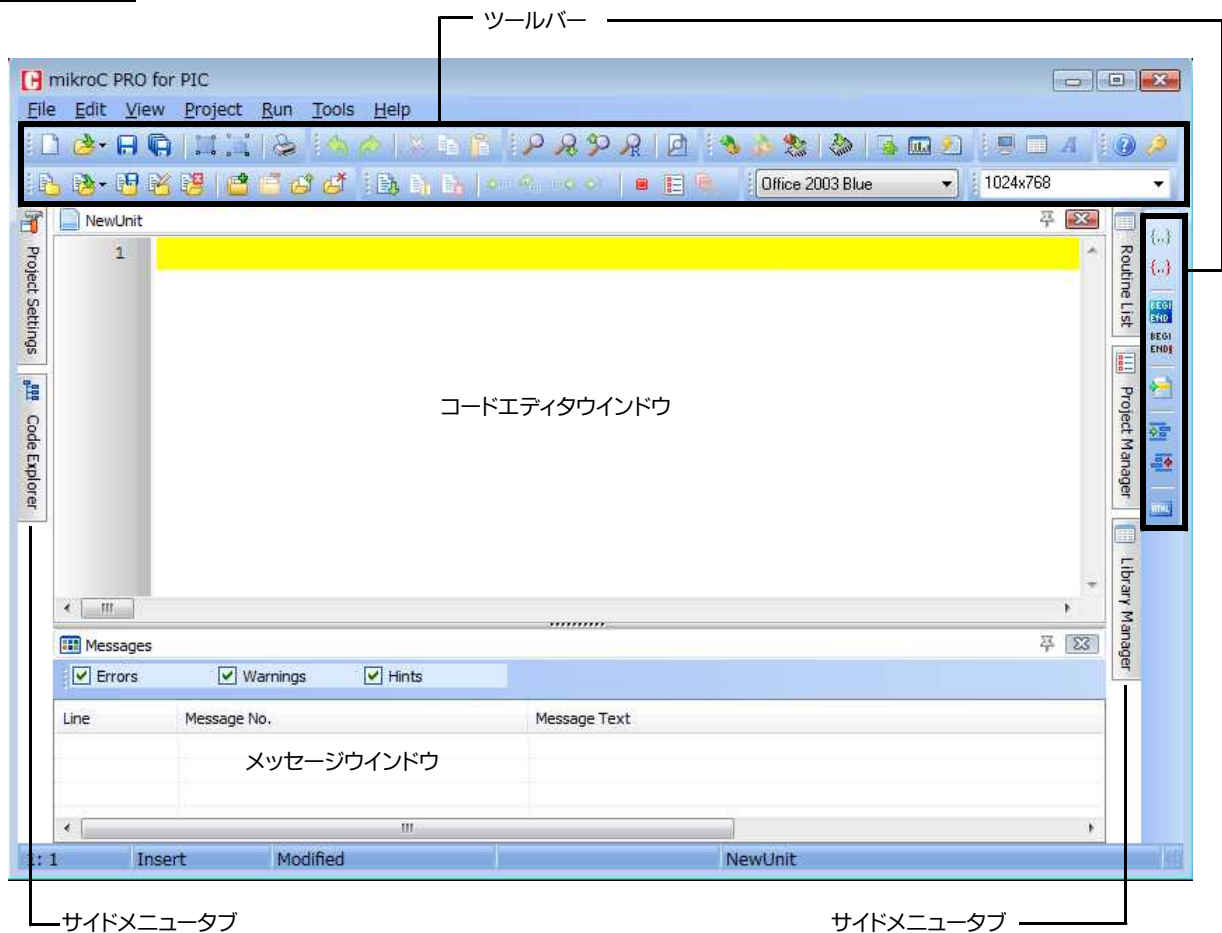
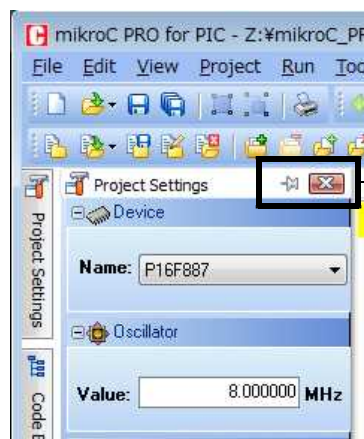
※実用的な大きさとしては、画面の解像度にもよりますが、14ポイント程度が見やすいです。

※右上にある"Font"ボタンではありませんので注意してください。

※フォントの種類は変更しないでください。

設定したら"OK"ボタンを押して完了します。さらにOptionsの画面では、"Apply"ボタンをクリックして設定内容を反映させた後、"OK"ボタンを押して完了します。

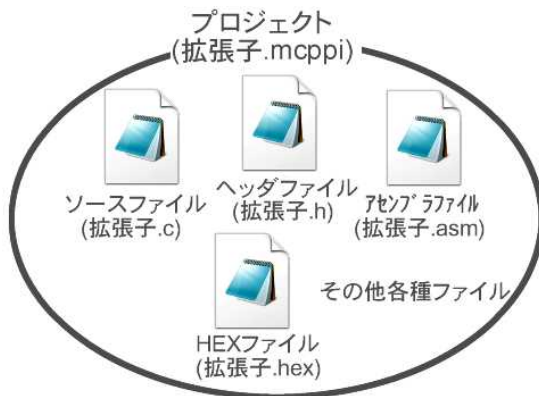
※その他にも詳細な設定が可能ですが、トラブルを避けるためフォントの変更以外は行わないことをお奨めします。

■mikroCの開発環境画面

サイドメニュータブは、便利な機能が割り当てられたタブです。クリックすると該当する画面がサイドから表示されます。デフォルト設定では、マウスのカーソルが上に乗っている時は、メニューは表示されたままとなり、カーソルを外すとメニューは自動的に隠れます。

左図は、"Project Settings"メニューの表示例です。
固定表示されたい場合には、☒ボタンとなりの マークをクリックすると固定されます。また強制的にメニューを隠したい場合は☒ボタンをクリックします。

チュートリアル①　～RB0の制御とボタン処理、変数の使い方～

チュートリアル①では最初の基本ということで、次のような簡単な内容のプログラムを作ってみます。

int型の変数を用意して、そこに入れた値を2進数でPORTBに出力するプログラムを作ってみましょう。変数を2つ用意して、外部のスイッチ入力によって、どちらの変数の値をPORTBに出力するのか指定できるようにしてみます。
デバイスは標準で付属しているPIC16F1939を、クロックは外部発振子の8MHzを内部のPLLで4倍して32MHz動作をさせてみます。

【1】プロジェクトの新規作成
プログラムを作り始める前に、プロジェクトを新規に作成します。mikroCでは1つのファームウエアに対して1つのプロジェクトとして、プロジェクト単位で管理します。プロジェクトには、これから作成するC言語のソースプログラム(拡張子.c)や、場合によって使用することとなるヘッダーファイル(拡張子.h)などのファイルが登録されます。その他、コンパイル後のHEXファイルなどもプロジェクトに含まれる1ファイルとして管理されます。

よくある間違いとして、プロジェクトの意味を理解していないと画面上に表示されているソースプログラムをコンパイルしているつもりなのに、実際には別のソースプログラムがコンパイルされていて思ったような動作をしなかったり混乱してしまったりというトラブルに繋がります。新しいプログラムを作るときには、必ずプロジェクトを新規に作成するようにしてください。

1　新規にプロジェクトを作成します。
メニューバーから"Project"→"New Project"をクリックします。
mikroCではウィザード形式でプロジェクトを新規に作成します。ウィザード画面が表示されたら、"Next"ボタンを押して続行します。

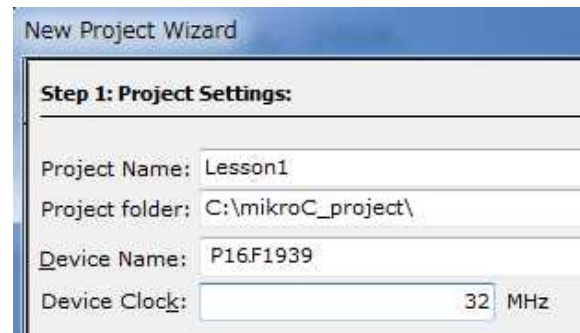
2　次の画面でプロジェクト名や使用デバイス名、動作周波数を入力します。まず最初にプロジェクト名を付けます。ここでは例として、"Lesson1"という名前にします。"Project Name"の所に"Lesson1"と入力します。

その下の"Project folder"の所にプロジェクトを保存するディレクトリを指定します。ここでは例として、Cドライブの直下に"mikroC_project"というフォルダを作り、そこにプロジェクトを作成することにします。"Browse"ボタンをクリックしてディレクトリを指定してください。

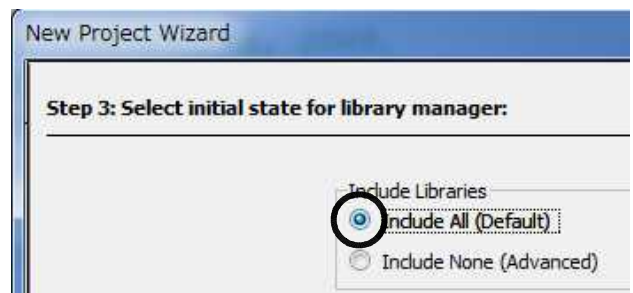
※プロジェクトを作成したフォルダには各種ファイルが作られますので、なるべく分かりやすいディレクトリにすることをお奨めします。

ディレクトリ名及びファイル名には、日本語や全角文字などの2バイト文字は使用できません。必ず半角英数字としてください。デスクトップやマイドキュメントなどは日本語が含まれてしまうことがあり、お奨めできません。

続いて、使用デバイスを選択します。"Device Name"のプルダウンから "P16F1939" を選択します。

クロック周波数を設定します。今回は、PICD-700SXに最初から付いている8MHzの水晶発振子を4逓倍して、32MHz動作で使用しますので、"Device Clock"の部分に、下図のように "32" と入力して32MHz に設定します。
小数点以下は自動的に0が挿入されますので、単に　32　とだけ入力してください。

"Next"ボタンを押します。

3　次の"Step2"はそのまま"Next"ボタンを押します。

4　続いて"Step3"の"Select initial state for library manager"では、プロジェクトに含める組込関数を「すべて」にするか「都度指定」にするかを指定します。ここではすべての組込関数を含めますので、"Include All(Default)"にチェックを入れて、"Next"ボタンを押します。

5　最後に"Finish"ボタンをクリックして完了します。

【2】プログラムの記述
　コードエディターウインドウにプログラムを記述しています。
プログラムを書く前にプログラムの概要を下記にまとめます。

　符号なし8ビットの変数aとbの2つを用意します。aとbにはあらかじめ値を代入しておきます。
　RA0とRA1をアクティブH（スイッチを押すと該当ピンがHレベルになるスイッチのこと）のスイッチ入力として使用し、RA0がHレベルになると変数aの値を、RA1がHレベルになると変数bの値を2進数としてPORTB(RB)に出力します。

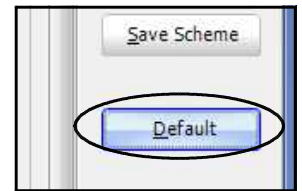
次のように記述しましょう。

```
void main(){

  unsigned int a,b;

  ANSELA = 0;
  ANSELB = 0;
  TRISA = 0x03;
  TRISB = 0;
  TRISC = 0;
  TRISD = 0;

  LATA = 0;
  LATB = 0;
  LATC = 0;
  LATD = 0;

  a = 100;
  b = 250;

  while(1){
     if (PORTA.B0 == 1) PORTB = a ;
     if (PORTA.B1 == 1) PORTB = b ;
  }
}
```

【3】コンフィギュレーションビットの設定
　プログラムが作成できましたので、続いてコンフィギュレーションビットの設定を行います。

　コンフィギュレーションビットとは、HEXファイル（C言語で作ったファイルをコンパイルした後にできあがる、最終的にPICマイコンに書き込むためのファイル）をPICに書き込む時にのみ設定できるハードウエアの基本的な動作条件を設定する設定項目のことをいいます。すごく重要な設定で、このコンフィギュレーションビットの設定が間違っていると、プログラムやハードウエアの設計が正しくてもプログラムが動かなかったり、期待した動作をしないなど・・・多くのトラブルの原因となります。

　コンフィギュレーションビットの設定は、プロジェクト単位で管理することができます。HEXファイル書き込み時にも都度設定することもできますが、いちいち設定するのは面倒なので、普通はプロジェクト単位で設定します。

　メニューバーの"Project"をクリックして"Edit Project"をクリックします。ダイアログが表示されます。
最初にデフォルト状態から設定を開始しますので、右下にある"Defailt"ボタンを押して下さい。

"Default"ボタンを押して、一度標準設定にした後、再度必要な項目を設定していきます。

　まず重要なのが発振子の種類です。PIC16F1939では大きく分けて内蔵発振子か外部発振子かを選ぶことになります。ここではボードに最初から装着されている8MHzの水晶発振子を使いますので、外部発振子の設定、さらに周波数が5MHz以上なので種類は"HS"になります。
※4MHzの時は"XT"にします。

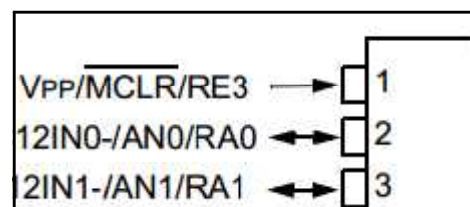
　外部発振子8MHzをもとにして内部でPLLで4逓倍して32MHzで動作させます。下記のように設定を変更してください。

・Oscillator Selection	HS Oscillator
・Power-Up Timer Enable	Enabled
・MCLR Pin Function Select	Disabled
・PLL Enable	Enabled
・Low-Voltage Programming Enable	Disabled

上記の設定以外はそのまま（デフォルト）の状態でかまいません。設定が正しく行われたことを必ずご確認下さい。
"OK"ボタンを押して元の画面に戻ります。

　この設定内容の意味ですが、まず発振子の種類は5MHzの外部発振子として"HS"を選択しています。

　Power Up Timerは、電源投入時に電源電圧の状態が安定するまでの期間プログラムを動作させないタイマーのことです。通常電源はONにした瞬間、過渡現象でふらつきが生じます。このふらついている時にプログラムが動くとプログラムが正しく動かなかったり誤作動することがありますのでそれを防止することができます。

　MCLR Pin Functionとは、MCLRピンの機能を設定する項目です。MCLRピンはPICマイコンのハードウエアリセットをかけるピンです。PIC16F1939の場合には1ピンがMCLRピンです。

上の図はPIC16F1939のデータシートに記載されているPICマイコンの図を抜粋したものです。1ピンはMCLRピンとRE3ピンを兼ねていることが分かります。"MCLR Pin Function"を有効(Enabled)にするとMCLRピンはリセットピンとなり、このピンをGNDに接地させるとPICマイコンにはハードウエアリセットがかかります。
"MCLR Pin Function"を無効(Disabled)にするとこのピンはRE3ピンとして使えます。ただし注意が必要なのはこのピンは入力ピンとしてしか使えないということです。データシートの図でも矢印が入力方向にしか付いていません。出力ピンとしては使えないので注意が必要です。

PLL Enableはその文字通り、PLLを有効にするかどうかです。PLLは逓倍回路のことでPIC内部にあります。ほとんどのPICでPLLは4逓倍となっており、クロック周波数を4倍にできます。例えば8MHzの外部発振子をPLLで4逓倍すれば32MHzで動作するということになります。

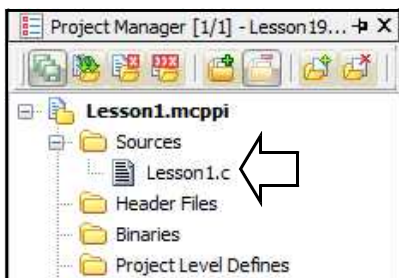
Low-Voltage Programming Enableはほとんどの場合、"Disabled"にしなくてはいけない項目です。PICマイコンはプログラムを書き込む際Vppピンに11V～13Vの高電圧を印加します。するとPICマイコンは書き込みモードだな、と判断しプログラムの書き込みができるようになります。Low-Voltage Programmingを有効にするとそういった高電圧を印加しなくてもプログラムの書き込みが可能となります。しかし本PICD-700SXをはじめ多くのPICマイコンライターはVppピンに高電圧を印加してプログラムを書き込みますのでここは無効(Disabled)にしておく必要があります。さらにここを不用意に有効にしてしまうと、PICマイコンが意図せず書き込みモードになり動作に不具合が発生することがありますので、通常この項目を見たら"Disabled"に設定するのが慣例となっています。

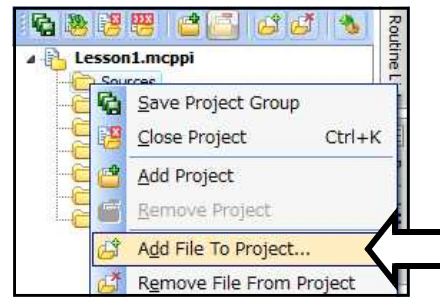
すごく大事なコンフィギュレーションビットの設定

PICの開発を始めると多くの方が戸惑うのがこのコンフィギュレーションビットの設定です。1つの原因は設定項目が英語だから。。もう1つはデバイス毎に設定項目が異なるからです。
プログラムが動かないと、「あれ?故障?もう故障!?」とすぐに機器やデバイスの故障を疑いたくなりますが、実際に機器やデバイスが故障することは滅多にありません。(使い方にもよりますが・・)
トラブルの最大の原因はコンフィギュレーションビットの設定が正しくないことです。コンフィギュレーションビットの設定がよく理解できていないと設定がおざなりになってしまい、デバイスを変えたりすると動かない・・・という原因になってしまうのです。
コンフィギュレーションビットの設定はちょっと複雑で難しいですが参考書やデータシートにも詳しく記載がありますので、ぜひそちらを読んでよく理解した上でご使用頂けることを願っております。

【4】プロジェクトへのファイルの登録確認

先にも説明しましたが、mikroCは、すべてのファイルをプロジェクトという単位で管理しています。よってプログラムを作成しても、そのソースファイル(拡張子.c)が、ファイルがプロジェクトに含まれていなければなりません。ビルドに先立って、必ず現在のプロジェクトにソースファイルが正しく登録されているか確認しましょう。

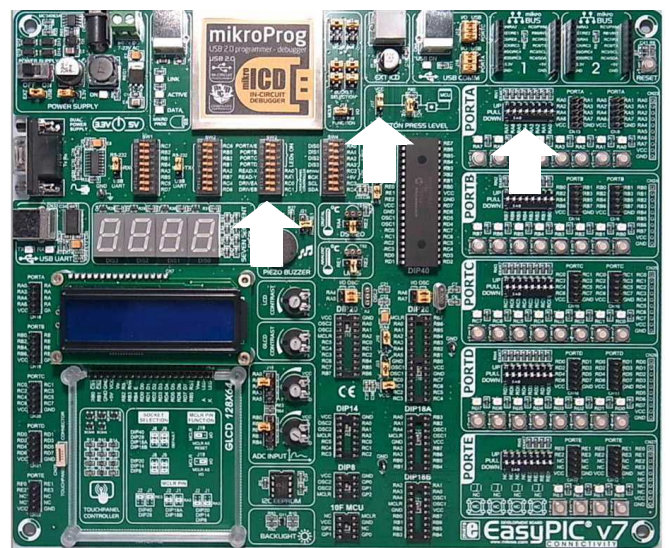
1 右サイドにあるサイドメニュータブの"Project Manager"をクリックします。現在作成しているプロジェクトに含まれる各ファイル類がツリー構造で表示されます。

"Sources"のツリーを展開し、ここに"Lesson1.c"が含まれていればそのままで続行できます。もし、何も登録されていない場合には、次の手順でソースプログラムを追加します。
※通常プロジェクトを新規で作成すると"プロジェクト名.c"のソースファイルが登録されます。

登録されていない場合は、"Sources"を右クリックするとメニューが表示されますので、その中から"Add File To Project"をクリックします。

ファイルを開くダイアログが表示されますので、先ほど作成したソースファイル "Lesson1.c"を選択して開きます。すると、プロジェクトにソースファイルが追加されます。

※このプロジェクトに含まれるファイルを確認する作業はとても重要です。mikroCは、どんな作業をする場合であっても必ずプロジェクト単位で行うため、画面上に表示されている、されていないに関わらず、必ずプロジェクトに含まれているファイルに対して操作が行われます。よって、プロジェクトに正しくソースファイルが登録されているかどうか確認しておかないと、結果として正しいHEXファイルを作れずに、トラブルの原因となってしまうことがあります。

【5】ハードウエアの物理的な設定

PICD-700SXボードの各種設定を行いましょう。
今回はRBはLEDに接続し、RAは定常時プルダウンに設定し、タクトスイッチが押された時に+5V(Vcc)に接続されるようアクティブHの設定にします。次のように設定してください。

①ディップスイッチSW3
→"PORTB"のスイッチのみON側にします。
②PORTAのディップスイッチ
→RA0とRA1のスイッチを"DOWN"側(下側)にセットします。
③ジャンパーJ17(タクトスイッチ設定)
→"Vcc"側(上側)をジャンパーソケットでショートします。

ボードの設定が正しくないとプログラムは期待通りに動作しません。物理的な状態が作成したプログラムの内容に矛盾しないように正しく設定する必要があります。

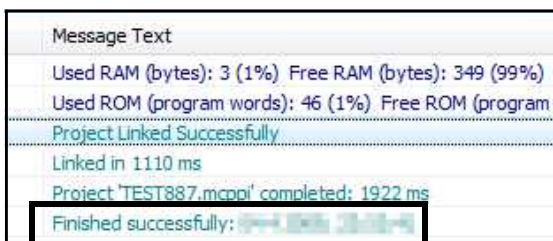
【6】ビルドと書き込みの実行

ビルドとは、C言語で記述したプログラムから、PICに書き込める形式のHEXファイルを作ることをいいます。PICにこのHEXファイルを書き込むと、PICが動作し始めるわけです。

mikroCと、PICD-700SXの書き込み用ソフトウエアmikroProg Suiteは連動させて動作させることができます。

1 ツールバーの"Build and Program"ボタンをクリックします。

なお、キーボードからは"Ctrl"+"F11"キーを押しても実行できます。

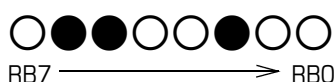
2 メッセージウインドウに進捗状況が表示されます。
エラーが発生すると、赤文字で表示されます。エラーがない場合には、"Finished Successfully"と表示され、すぐに書き込みソフトウエアであるmikroProg Suiteが起動します。

mikroProg Suiteが起動した後、下図のようなメッセージが表示された場合には、デバイスの設定が正しく行われていませんので、デバイスの種類の設定を再度ご確認下さい。

書き込みが終わるとプログラムがPICD-700SX上で動きます。

【7】動作確認

書き込みが完了するとプログラムがPICD-700SXボード上ですぐに動きます。動作を確認してみましょう。

PICD-700SXのタクトスイッチ群のなかにあるRA0のスイッチを押します。RBに接続されたLEDが点灯します。RA0なので変数aの値、ここでは100が2進数でLEDに現れます。

100は2進数表記では　01100100　となります。
RBのLEDは下記のように点灯しているはずです。

※○は消灯、●は点灯

同様にして、RA1のスイッチを押すと変数bの値、すなわち250が2進数としてRBのLEDに表示されます。

ここまでで「プロジェクトの作成」→「プログラムの作成」→「ビルド及び書き込みの実行」という一連の作業はできたことになります。
どんなプログラム作成でもこの一連の操作の流れはほぼ同じ場合が多いですので、よくご理解頂き次のチュートリアルもお試し下さい。

PORTBのLED全体がボーッと薄く点灯する場合には、PORTBのディップスイッチをすべてPull-Down側にセットしてください。
PORTBにはLCDが接続されており、その関係でLEDが薄く点灯することがあります。

■プログラムの解説

void main(){…}
main関数です。C言語では1つのプログラム内に必ず1つのmain関数が必要です。voidはmain関数の型で戻り値がない関数であることを宣言しています。{ }で囲まれた範囲がmain関数の範囲となります。

unsigned int a , b ;
符号なしのint型変数を宣言しています。C言語では文の最後には必ず ; （セミコロン)をつけます。
符号なしのint型の場合扱える値は、0～65535までです。

ANSELA = 0;
ANSELB = 0;
PIC16F1939では、PORTAとPORTBはADコンバータのアナログ入力ピンとしてアサインされています。デジタルI/Oとしてピンを使用する場合にはANSELxレジスタに0を代入して、デジタルモードに設定します。ANSELAレジスタはPORTAの、ANSELBはPORTBのADコンバータについて設定をするレジスタです。
電源投入時はすべてビットが1になっており、アナログ入力ピンに設定されています。よって、デジタルI/Oピンとして使用する場合には、プログラムの中で必ず両レジスタに0を代入してデジタルモードに設定しないと期待通りにプログラムが動かなくなってしまいます。

TRISA = 0x03;
TRISB = 0;
TRISレジスタは、ピンの入出力方向を設定するレジスタです。
該当のビットが0の場合は出力、1の場合は入力となります。本チュートリアルではRBは全ピン出力、RAはRA0とRA1が入力なので0x3として設定しています。
C言語では16進数の値を表現する時は数値の前に0xを付けます。

PORTB = 0 ;
PORTBに0を代入しています。

LATA = 0;
LATB = 0; ...
LATxレジスタは、もともとPIC18シリーズから搭載されたレジスタですが、PIC16F19xxシリーズにも搭載され、やや使い方がややこしくなり、このレジスタに存在に疑問を抱かれる方も多いようです。

LATxレジスタはPORTxレジスタと似ていますが実際には、PORTxレジスタの値、すなわちI/Oピンへの出力は実際には、LATxレジスタへの出力となります。この出力の内容が、実際のI/Oピンに反映されるかどうかは、TRISxレジスタへの設定で決まります。
また、大きく注意が必要なのはI/Oピンの状態をプログラムから読

み込む際、PORTxレジスタからの読み込みを行うと、実際のI/Oピンの現在の入力状態を読み込むことができます。
一方LATxレジスタからの入力については、実際のピンの状態を反映しているわけではなく、単に現在のデータラッチ状態を入力するだけということに注意が必要です。
分かりにくい場合には、I/Oピンへの出力はLATxレジスタ又は、PORTxレジスタに値を代入し、I/Oピンの状態を読み込みたい場合には、PORTxレジスタから読み込むとよいとまずは覚えておきます。

a = 100 ;
b = 250 ;
int型変数に値を代入しています。

while(1){…}
while文は( )内の値が1又は真であれば{ }内の処理を繰り返す繰り返し文です。ここでは評価式を1としていますので永久に{ }内が実行されます。
C言語では処理を永久ループさせたい場合にはよくwhile文を使用します。

if (PORTA.B0 == 0) PORTB = a ;
if文を使用した入力判定部分です。
( )内の評価式が真の時にその後ろに記述された文が実行されます。
mikroCでは「レジスタ名.Bビット数」でレジスタの1つのビットを指定できます。ここではRA0を指しています。
==(イコール2つ)は比較演算子の「等しい時」を表す演算子です。
よくある間違いとして=を1つしか記述しない場合があります。=が1つの場合には代入演算子となってしまい正しく動作しません。IF文の時には==(イコール二つ)と覚えておくと間違いを減らせます。
PORTB = a でPORTBレジスタに変数aの値を代入しています。その結果がLEDで表示されます。

チュートリアル② ~LCDの使用と、配列・ポインタ~

PICD-700SXに装着されている2行16文字のLCDに文字列を表示します。C言語では、文字列を扱う場合配列を使用します。
本チュートリアルでは固定した文字列をLCDに表示させてみましょう。
本チュートリアルから先のすべてのチュートリアルは、動作周波数を8MHzにします。32MHz動作をさせるほどの高速クロックが要求されるようなプログラムではありませんので8MHz動作とします。

■プロジェクトの新規作成

1 新規にプロジェクトを作成します。
メニューバーから"Project"→"New Project"をクリックします。
mikroCではウィザード形式でプロジェクトを新規に作成します。ウィザード画面が表示されたら、"Next"ボタンを押して続行します。

2 次の画面でプロジェクト名や使用デバイス名、動作周波数を入力します。まず最初にプロジェクト名を付けます。ここでは例として、"Lesson2"という名前にします。"Project Name"の所に"Lesson2"と入力します。
その下の"Project folder"の所にプロジェクトを保存するディレクトリを指定します。"Browse"ボタンをクリックしてディレクトリを指定してください。
続いて、使用デバイスを選択します。"Device Name"のプルダウンから "P16F1939" を選択します。
クロック周波数を設定します。今回は、PICD-700SXに最初から付いている8MHzの水晶発振子を使用しますので、"Device Clock"の部分に、下図のように "8" と入力して8MHz に設定します。
"Next"ボタンを押します。

3 次の"Step2"はそのまま"Next"ボタンを押します。

4 続いて"Step3"の"Select initial state for library manager"では、プロジェクトに含める組込関数を「すべて」にするか「都度指定」にするかを指定します。ここではすべての組込関数を含めますので、"Include All(Default)"にチェックを入れて"Next"を押します。
最後に"Finish"ボタンをクリックして完了します。

■プログラムの作成

```
sbit LCD_RS at RB4_bit;
sbit LCD_EN at RB5_bit;
sbit LCD_D4 at RB0_bit;
sbit LCD_D5 at RB1_bit;
sbit LCD_D6 at RB2_bit;
sbit LCD_D7 at RB3_bit;

sbit LCD_RS_Direction at TRISB4_bit;
sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D4_Direction at TRISB0_bit;
sbit LCD_D5_Direction at TRISB1_bit;
sbit LCD_D6_Direction at TRISB2_bit;
sbit LCD_D7_Direction at TRISB3_bit;

void main() {
  char txt_a[] = "Hello";
  char txt_b[] = "World";

  ANSELA = 0;
  ANSELB = 0;

  Lcd_Init();
  Lcd_Cmd(_LCD_CLEAR);
  Lcd_Cmd(_LCD_CURSOR_OFF);

  Lcd_Out(1,1,txt_a);
  Lcd_Out(2,1,txt_b);
}
```

■コンフィギュレーションビットの設定
プログラムが作成できましたので、続いてコンフィギュレーションビットの設定を行います。
メニューバーの"Project"をクリックして"Edit Project"をクリックします。ダイアログが表示されます。
最初にデフォルト状態から設定を開始しますので、右下にある"Defailt"ボタンを押して下さい。

外部発振子8MHzで動作させます。下記のように設定を変更してください。

・Oscillator Selection　　HS Oscillator
・Power-Up Timer Enable　　Enabled
・MCLR Pin Function Select　　Disabled
・PLL Enable　　Disabled　★
・Low-Voltage Programming Enable　　Disabled

上記の設定以外はそのまま(デフォルト)の状態でかまいません。チュートリアル①と異なり、"PLL Enable"の設定が"Disabled"になっていることに注意して下さい。
　設定が正しく行われたことを必ずご確認下さい。"OK"ボタンを押して元の画面に戻ります。

■プログラムの書き込みと実行
　続いて、ビルドと書き込みを行ってみましょう。プログラムが書き込み終わると、LCDの1行目に"Hello"と表示され、2行目に"World"と表示されます。

■プログラムの解説

```
sbit LCD_RS at RB4_bit;
sbit LCD_EN at RB5_bit;
sbit LCD_D4 at RB0_bit;
sbit LCD_D5 at RB1_bit;
sbit LCD_D6 at RB2_bit;
sbit LCD_D7 at RB3_bit;

sbit LCD_RS_Direction at TRISB4_bit;
sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D4_Direction at TRISB0_bit;
sbit LCD_D5_Direction at TRISB1_bit;
sbit LCD_D6_Direction at TRISB2_bit;
sbit LCD_D7_Direction at TRISB3_bit;
```

　main関数の前にsbitタイプを用いてLCDの物理的な配線状態を定義します。
　sbitタイプは、PICマイコンのSFR(レジスター)の特定のビットに対してアクセスするため機能を提供します。
　ここでは、その1つの使い方としてLCDの接続状態をあらかじめ定義しています。sbitは、TRISレジスタに対しても特別な書式として記述することができます。LCD関連のライブラリを使用する場合には、main関数の前で、この定義を記述してください。

　※この定義をmain関数の中で記述してしまうとエラーになります。

char txt_a[] = "Hello";
　char型配列を宣言しています。char型は文字を扱う8ビットの型です。C言語では文字はASCIIコードにて扱われます。
　通常配列を宣言する際には要素数(インデックス値)を[　]内に入力しますが、[　]内を空欄にすると自動的に要素数が設定されます。配列や変数を宣言する際に初期値を代入できますが、char型に限り文字列で初期化できます。"(ダブルクォーテーション)で囲むと文字列として認識されます。

Lcd_Init() ;
　LCDを初期化する関数です。LCDを使用する前に記述します。

Lcd_Cmd(_LCD_CLEAR);
　Lcd_Cmd(　)関数は、LCDの制御コマンドを送信する関数です。
　ここでは、画面をクリアーするコマンドと、カーソル(描画位置に点滅するカーソル)を表示しない設定にしています。
　※mikroCのマニュアルは、"Help"→"Help"で見ることができます。

Lcd_Out(1,1,txt_a) ;
　Lcd_Outは4ビットモードでLCDに文字や文字列を表示する時に使用します。書式は下記の通りです。

　Lcd_(行, 列, char *text);

　*textは、char型のポインタという意味です。ここでは、配列名を記述することで配列の0番目の要素のアドレス値を指定しています。

■配列とは?
　配列は、連続したメモリー領域にスペースを確保する仕組みです。通常、変数を宣言するとメモリー上に格納スペースがコンパイラーによって割り当てられます。しかしこの作業はコンパイラーが自動的に行うものであり、開発者はどの変数がメモリー上の何番地に割り当てられたからは分かりません。
　文字列のように関係性が連続的なデータの場合、データは連続するアドレスに配置される必要があります。そこで変数ではなく配列を宣言します。配列を宣言するとコンパイラーは、連続したメモリー領域を確保します。配列は下記の書式により宣言できます。

```
型　配列名[要素数] = {初期値0,初期値1,・・・} ;
```

　初期値は省略できます。char型配列の場合のみ文字列で初期化することができます。要素数を省略すると、自動的に要素数が決まります。
　配列名を指定すると、その配列の要素数0のアドレス値が参照されます。要素数0のアドレスが分かれば、あとのデータは連続するアドレスに格納されていますので、配列のサイズさえ分かれば、データの連続性を損なわずにデータを取り出すことができます。

■ポインタとは?
　ポインタは、通常の変数と異なり格納されている値を参照するのではなく、その変数に割り当てられているメモリー領域上のアドレスを参照する仕組みです。
　例えば、 int a = 30;　と宣言した場合、int型変数aは、初期値に30が代入された状態で、メモリー領域上のどこかに配置されます。しかし通常開発者は、変数aを参照しても30という代入された値は分かりますが、変数aがメモリー上の何番地に配置されているのかは分かりません。
　ポインタ変数は、ある変数のアドレス値だけを専門に扱う変数です。C言語では、ポインタ変数を宣言する場合には、変数名の前に*(アスタリスク)を付けます。また、ある変数のアドレス値を取得したい場合には、変数の前に&演算子を付けます。下記に例を示します。

```
int a, b ;    }①
int *p ;

p = &a ;       ②

*p = 100 ;    }③
b = *p + 1 ;
```

　①は、変数の宣言です。ポインタ変数pはint型変数のアドレス値を扱うためのものです。
　②で、ポインタ変数pに変数aのアドレス値を代入しています。この時、

ポインタ変数pには*が付かないことに注意します。
③ではポインタ変数pが指し示すアドレスにある変数、つまりは変数aに100を代入しています。ポインタ変数が指し示す先のアドレスの変数に値を代入する場合には、ポインタ変数に*を付けます。
さらに、ポインタ変数の指し示す先の変数、すなわち変数aの値に1を加算していますので、変数bの値は、101ということになります。

なぜこのように複雑な方法を使うのでしょうか。
ポインタ変数は、配列と組み合わせると色々な応用ができます。配列のように連続したアドレスに値が格納されている場合、ポインタで参照すれば、ポインタの値を増やしたり減らしたりするだけで、値を取り出すことができます。例えばシリアル通信で連続したデータを受信した場合、ポインタで先頭のアドレスを参照してあとはポインタの値をインクリメントするだけで連続したデータを参照できます。

チュートリアル③　～ADコンバータとTMR0割り込み～

■プログラムの内容

PICD-700SXに搭載の2行16文字のLCDに、数字をカウントアップ表示するカウントアップアプリケーションを作ります。カウントアップの時間間隔は、ADコンバータで取得した値を参照し、0～1023段階でカウントアップの間隔を調整できるようにします。

RC0をアクティブHの入力として、スイッチを押した時からカウントを開始するようにします。RC1もアクティブHとして、スイッチを押すと、カウントアップを停止するように作ります。

時間の生成はTMR0の割り込みを使用します。割込プログラムの作成とADコンバータの利用、変数の値をLCDに表示する方法などを紹介します。

■プログラムの仕組みを考えましょう

数値は0～9999までとします。カウントは0から始まり、9999まで達すると0に戻ります。符号無しint変数に値を入れて、その値をLCDに表示させます。

ADコンバータは今回はAN2(AN2はRA2にアサイン)の入力電圧をサンプリングし、その値を変数に代入します。分解能は10ビットなので、0V～+5Vの電圧が印加されると、0～1023までの値となって、値が変数に代入されます。その値を参照してカウントアップの判定を行います。

時間はTMR0のオーバーフロー割込を使います。今回はTMR0を256μ秒毎にオーバーフローさせます。256μ秒毎に割込が発生しますので、その割込回数と先のADコンバータの値を比較することで、カウントアップの時間を調整します。TMR0のオーバーフロー割込は電源投入時又はリセット時は無効にしておき、RC0がHレベルになった時から開始します。RC1がHになったらカウントアップを停止します。

PIC16F1939のバグについて

PIC16F19xxデバイスには内蔵ADコンバータのバグがあることが分かっておりシリコンエラッタがリリースされています。このバグが修正されていないデバイスの場合、下記のADコンバータプログラムを実行するとプログラムが動かなくなります。
その場合には、PIC16F19xxのADC変換完了フラグに依存しない"ADC_Get_Sample関数"を使うと問題を回避できます。詳しくは下記のFAQをご覧ください。

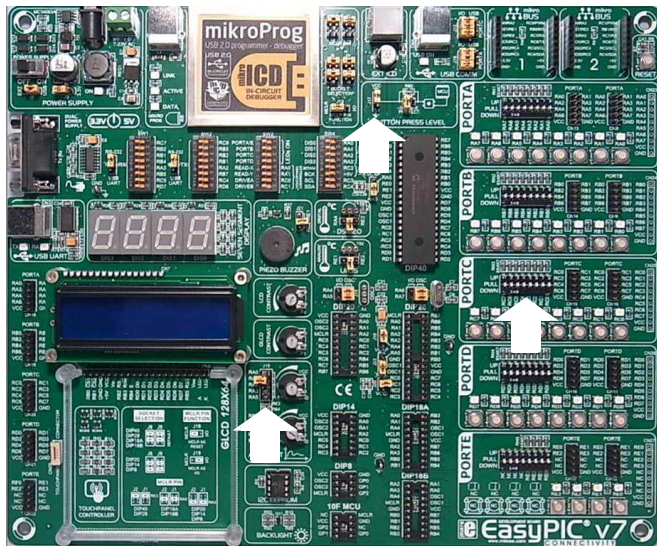
http://www.microtechnica.tv/faq/faq.cgi?kate=picdev&faq=3

■プログラムの作成

```
sbit LCD_RS at RB4_bit;
sbit LCD_EN at RB5_bit;
sbit LCD_D4 at RB0_bit;
sbit LCD_D5 at RB1_bit;
sbit LCD_D6 at RB2_bit;
sbit LCD_D7 at RB3_bit;
sbit LCD_RS_Direction at TRISB4_bit;
sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D4_Direction at TRISB0_bit;
sbit LCD_D5_Direction at TRISB1_bit;
sbit LCD_D6_Direction at TRISB2_bit;
sbit LCD_D7_Direction at TRISB3_bit;

unsigned int ad_res=0,total=0,intrr=0;

void init(){
  ANSELA = 0x04;
  ANSELB = 0;
  ADCON0 = 0x09;
  ADCON1 = 0x80;
  TRISC = 3;
  OPTION_REG = 0x80;
  TMR0 = 0;
  INTCON = 0;
  Adc_Init();
  Lcd_Init();
  Lcd_Cmd(_LCD_CLEAR);
  Lcd_Cmd(_LCD_CURSOR_OFF);
}

void interrupt() {
  intrr++;
  if(ad_res < intrr){
    if(total<9999){
      total++;
    }else{
      total=0;
    }
    intrr=0;
  }
  TMR0   =   0;
  INTCON = 0x20;
}

void main() {
  char txt[7];
  init();

  total=0;  intrr=0;

  while(1){
    if(PORTC.B0==1)INTCON = 0xA0;
    if(PORTC.B1==1)INTCON = 0;
    IntToStr(total,txt);
    Lcd_Out(1,1,txt);
    ad_res = ADC_Read(2);
  }
}
```

■PICD-700SXボードの設定

このプログラムを正しく動作させるため、下記のようにボードの各種設定をセットしてください。

・PORTCのディップスイッチをすべて"DOWN"側に設定
・上記以外のディップスイッチはSW4の6番以外はすべてOFFに設定
・J17をVCC側に設定
・J15(ADコンバータ用設定)のジャンパーをRA2の位置に設定

上記の設定で、PORTCは定常時は常にL、スイッチを押すとHになります。ADコンバータはAN2(RA2)が入力として使用され、P1のボリュームを回転させると0V～+5Vの電圧がRA2に印加されます。

■プログラムの動作確認

プログラムを書き込んで動作を確認してみましょう。書き込み直後はLCDに"0"と表示されています。RC0のスイッチを押すとカウントアップが開始されます。P1ボリュームを左側に回転させるとカウントアップが速く、右に回転させるとカウントアップが遅くなります。

RC1のスイッチを押すとカウントを停止し、再度RC0のスイッチを押すとカウントアップが開始されます。

■プログラムの解説

```
unsigned int ad_res=0, total=0, intrr=0;
```

この3つの変数は、グローバル変数と呼ばれる変数の宣言方法です。変数を関数の中に記述すると、その変数はその変数内だけで使用できる変数(これをローカル変数と言います)になりますが、このようにプログラムの先頭部分(main関数の外)で宣言すると、その変数はどの関数からも使用できるグローバル変数となります。
ここでは、3つの変数を宣言しており、初期値としてすべての変数に0を代入しています。変数宣言時に代入演算子で値を初期値として代入することができます。

```
void init(){
```

マイコンの初期値を設定するために関数を作っています。ここではinitという名前の関数を作っています。
voidとは戻り値のない関数という意味で、initの後ろの( )に何も記述されていないのは引数がない関数ということを意味します。
すなわち、この関数は呼び出されたらその内容を実行後、戻り値を持たずに呼び出された場所に戻る関数であることを意味しています。

```
ANSELA = 0x04;
ANSELB = 0;
ADCON0 = 0x09;
ADCON1 = 0x80;
```

ADコンバータを使用するためのレジスタの設定です。
ANSELAレジスタはAN0～AN7のどのチャンネルをアナログ入力にするかを決めるレジスタです。今回はAN2をアナログ入力にしますので、ANSELAレジスタに0x04を代入しています。(該当ビットが1の時アナログ入力ピンとなります。)

ADCON0レジスタは、ADコンバータ用のクロック源の選択及びチャンネル選択、ADコンバータ機能の有効/無効設定のレジスタです。チャンネルはAN2を選択し、ADコンバータの機能を有効に設定しました。
ADCON1レジスタは、ADコンバータからの10ビットの戻り値を左詰めにするか、右詰めにするかの設定と、基準電圧の設定を行います。mikroCではADコンバータの結果は右詰めとします。
基準電圧は、VssとVddに設定しています。(PICの電源電圧の0V～+5Vが基準電圧となります)

```
OPTION_REG = 0x80;
```

OPTIONレジスタの値をセットしています。
本プログラムでは、TMR0のクロック源は内部クロックにして、周波数を分周するためのプリスケーラーはTMR0に使用し、その分周比は1:2に設定しています。

TMR0のオーバーフロー周波数は下記の式で計算できます。
「オーバーフロー周波数＝システムクロック÷4÷プリスケーラ分周比÷256」

システムが8MHzのシステムクロックで動作している場合には上式で計算すると、約3.9KHzとなります。逆数を取ると約256μ秒となり約256μ秒ごとに割込が発生することになります。
割込が発生すると、void interrupt()が実行されます。

```
INTCON = 0;
```

INTCONレジスタは割込についての管理を行っているレジスタです。0を代入することで割込を無効にしています。RC0がHになるまで、カウントアップは開始させないので、初期状態は0として割込を発生させないようにしています。

```
INTCON = 0xA0;
```

INTCONレジスタは、各種割込の設定を行うレジスタです。
割込を有効に設定し、TMR0の割込を有効にしています。
TMR0は8ビットのカウンタで、0～255までカウントしますが、255から0に戻る際にオーバーフローが発生します。オーバーフローが発生するとINTCONレジスタのビット2が1になります。これにより割込の発生をソフトウエア側で知ることができます。
なお、このINTCONレジスタのビット2は、ソフトウエア側で手動で0に戻す必要があります。

```
void interrupt() {
```

mikroCでは割込が発生すると、必ずこの関数が呼び出されます。main関数実行中に割込発生で直ちにこの関数が呼び出され、その内容が実行されます。割込ルーチン実行中は別の割込は発生しません。

```
intrr++;
```

割込の発生回数をカウントする変数です。++演算子はインクリメント演算子と呼ばれ、変数に付けることで1ずつ値を増加(インクリメント)させます。割込が発生する毎にこの変数の値が1ずつ増加します。

if(ad_res < intrr){

ad_res変数はADコンバータの値を格納している変数です。このADコンバータで取得した値と、割込の回数を比較することで、割込回数の方が少なければカウントアップはせずに割込処理を完了します。ADコンバータの値が割込回数より少なくなった場合にはカウントアップをさせます。この仕組みにより、ADコンバータの値によって、カウントアップの速度が変わる仕組みになっています。

if(total<9999){

9999に達したら0に戻す必要がありますので、if文で判定しています。9999に達する前はtotal++で、変数totalの値をインクリメントし、カウントアップさせ、9999に達したら、else以降の文が実行され、ここではtotal=0で変数が0に初期化されます。

TMR0 = 0;
INTCON = 0x20;

TMR0を0に戻し、INTCONレジスタの値を再度0x20に初期化しています。INTCONレジスタはTMR0の割込発生後、オーバーフローを通知するビット2が1となり、このビットはソフトウエア側でリセットする必要があります。

init();

先ほど記述した初期化の文をまとめて記述したinit関数をコールしています。

if(PORTC.B0==1)INTCON = 0xA0;

RC0の値を判定します。RC0がHで、INTCONレジスタの値が0xA0になります。これにより、割込が有効となり、TMR0がオーバーフローすると、割込ルーチンが実行されて、カウントアップが開始されます。

if文は条件式が真の時実行されるルーチンが長い場合には{ }でくくりますが、1つしかない時などは、{ }でくくらずに記述することができます。

IntToStr(total,txt);

型変換です。C言語では型の扱いが厳密であるため、変数や関数を使用する場合には常に型を意識しなければなりません。

変数totalは、int型変数ですが、このint型変数はLcd_Out関数の引数としては指定できません。Lcd_Out関数で出力する文字列を扱う場合には、必ずchar型ポインタ変数でなくてはなりません。char型のポインタ変数は、char型配列の先頭アドレスです。mikroCには、各変数の型からstring(文字列形式)にする関数があります。

IntToStr関数は、int型変数の値を文字列形式に変換します。文字列形式とは、すなわちchar型の配列です。char型のtxt配列を宣言して、そこに値を代入しています。なお、配列の大きさは型変換関数によって定められており、IntToStr関数では要素数は7にすることが決まっています。

ad_res = Adc_Read(2);

Adc_Read()関数は、( )内に指定したチャンネルからAD変換の値をunsigned型で取得し、変数に代入します。

チュートリアル④ ～UART通信と7セグメントLEDの駆動～

UARTは非同期式シリアル通信で、クロック信号と同期させずに時間的なタイミングのみで通信を行う通信方式です。パソコンのRS232CはこのUART通信の1つです。クロック信号がないため、送信側と受信側で通信速度をあらかじめ決めておくことで、時間間隔によってデータを送受信します。

7セグメントLEDの駆動にはダイナミック点灯方式を使います。
ダイナミック点灯は、人間の目の残像現象を利用して駆動する方法です。(ダイナミック点灯方式について詳しく知りたい方はインターネットで検索するとたくさん紹介が出てきます。)

■プログラムの内容

このチュートリアルでは、ASCII文字で0000～9999までの文字列をUART通信で受信し、その値を7セグメントLEDに表示させるプログラムを作ります。UARTの通信速度は115200bpsとし、PIC16F1939は8MHzの発振子で動作させることにします。

ダイナミック点灯の駆動にTMR0の割込を使用し、コモンピンを駆動します。UART通信についてもいつ信号が送られるか分からないため、UART受信割込を使用します。2つの割込を同時に使用するため、その方法もこのチュートリアルで学習できます。

■7セグメントLEDの仕組み

PICD-700SXでは、7セグメントLEDの各セグメントはPORTDに接続されています。また7セグLEDのコモン端子はトランジスターを介してDIS0～DIS3(各桁)がそれぞれRA0～RA3に接続されています。いずれも該当のピンをHレベルにすると点灯します。数字の形成は次のページの通りとなります。

7セグ側	PIC側の結線先
a	RD0
b	RD1
c	RD2
d	RD3
e	RD4
f	RD5
g	RD6
dp	RD7

数字を形成する場合にはPORTDの次のビットをHighレベルにします。

表示数字	Hレベルにするビット	PORTDの値(16進)
0	RD0、RD1、RD2、RD3、RD4、RD5	0x3F
1	RD1、RD2	0x06
2	RD0、RD1、RD2、RD3、RD4、RD6	0x5B
3	RD0、RD1、RD2、RD3、RD6	0x4F
4	RD1、RD2、RD5、RD6	0x66
5	RD0、RD2、RD3、RD5、RD6	0x6D
6	RD0、RD2、RD3、RD4、RD5、RD6	0x7D
7	RD0、RD1、RD2、RD5	0x27
8	RD0、RD1、RD2、RD3、RD4、RD5、RD6	0x7F
9	RD0、RD1、RD2、RD3、RD5、RD6	0x6F

最初に7セグLEDのDIS0に表示する数値をPORTDに出力して、RA0をHに、他をLとします。するとDIS0には数字が表示されますが、DIS1～DIS3は消灯したままなので、数字は表示されません。

続いて、DIS1に表示する数値をPORTDに出力して、RA1だけをHにします。すると今度はDIS1にだけ数字が表

示され他の桁は消灯しています。この方法でDIS0～DIS3までを約5ミリ秒程度で繰り返します。

本当は常に1桁だけの数字が点灯しているのですが、高速に繰り返しているので人間の目には残像となって映るので4桁とも数字が表示されているように見えるのです。

もし仮に1桁ずつ制御した場合（スタティック点灯方式）、7本×4桁=28で合計28本のI/Oピンを使わないと制御できません。しかしダイナミック点灯であれば、7本+4桁=11で合計11本のI/Oピンで制御ができ、ずっと効率よく回路を構成できます。

ただしダイナミック点灯の欠点は常にコモン端子の制御を行っていないと表示がおかしくなってしまうことです。PICのようなシングルタスクのMPUでは、7セグLEDの処理に追われていると他の処理をするのが大変です。

また、他の処理で遅延が発生すると表示が崩れたり、ちらつきを感じてしまいます。

そこで割込をうまく使ってやる必要があります。プログラムでは、このダイナミック点灯の約5ミリ秒のコモン端子の制御にTMR0割込を使います。

■プログラムの作成

```
unsigned short num[4];
unsigned short digit=0, cycle=1, uart_data;

void init(){
  ANSELA = 0;
  ANSELB = 0;
  TRISA = 0;
  TRISB = 0;
  TRISC = 0;
  TRISD = 0;
  INTCON = 0b11100000;
  OPTION_REG = 0b10000100;
  PIE1 = 0b00100000;
  PIR1.B5 = 0;

  UART1_Init(115200);

  return;
}

void interrupt() {
     unsigned short i;

     if(INTCON.TMR0IF == 1){
         switch (num[digit]){
             case 0: PORTD=0x3F; break;
             case 1: PORTD=0x06; break;
             case 2: PORTD=0x5B; break;
             case 3: PORTD=0x4F; break;
             case 4: PORTD=0x66; break;
             case 5: PORTD=0x6D; break;
             case 6: PORTD=0x7D; break;
             case 7: PORTD=0x07; break;
             case 8: PORTD=0x7F; break;
             case 9: PORTD=0x6F; break;
         }

         PORTA = cycle;
         cycle = cycle << 1;
         if(cycle>0b00001000) cycle=1;
         digit = digit + 1;
         if(digit==4) digit=0;
     }

     if(PIR1.RCIF == 1){
        for(i=0;i<4;i++){
            num[i]=0;
        }

        for(i=3;i>=0;i--){
           while(!UART1_Data_Ready());
           uart_data = uart1_read();
           num[i] = uart_data-0x30;
           if (uart_data == 0x0D)break;
        }
     }

     INTCON = 0xE0;
     PIR1.B5 = 0;
     RCSTA.B4 = 0;
     RCSTA.B4 = 1;
     TMR0  =  0;
}

void main() {

   init();

}
```

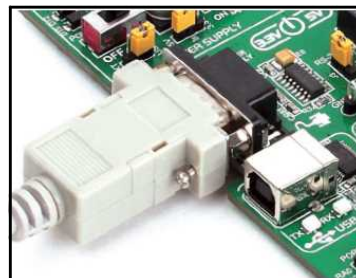
■PICD-700SXボードの設定

このプログラムを正しく動作させるため、下記のように設定します。

・SW1の"RC7"(1番)をONに設定、それ以外はOFFに設定
・SW2の"RC6"(1番)をONに設定、それ以外はOFFに設定
・SW4の1番～4番のスイッチをONに設定、それ以外はOFFに設定
・LCDは本体から取り外してください。

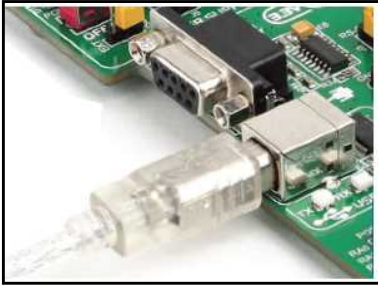
また、RS232C通信についてPICD-700SXでは従来のD-SUB9ピンタイプのポートを使うか、USB-UART変換ICを介してUSBポートでUART通信を使うかを選択できます。次のようにどちらかを選んで配線し、ボードの設定をしてください。

(1)D-SUB9ピンのRS232Cポートを使う場合

①パソコンのRS232Cポートと、PICD-700SXのRS232Cポートを全結線ストレートケーブルで接続します。RS232Cポート同士をRS232Cケーブルで接続してください。

②PICD-700SXのディップスイッチSW1及びSW2を正しく設定します。SW1のRC7とSW2のRC6をON側にし、その他をOFF側にセットしてください。

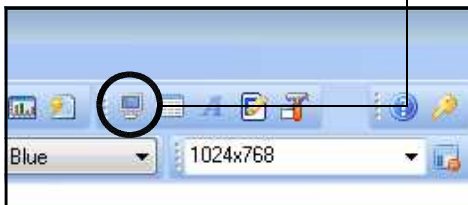
③ジャンパーソケットJ3とJ4を設定します。J3及びJ4いずれも上側の"RS232C"と書かれた方にソケットをセットします。

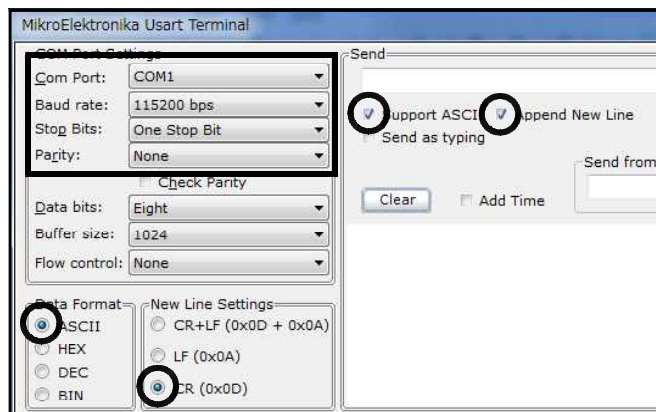
(2)USBポートで仮想COMポートを使う場合

①パソコンのUSBポートと、PICD-700SXの実験用USB-UART変換USBポート(PICD-700SX基板左側面)をUSBケーブルで接続します。

②PICD-700SXのディップスイッチSW1及びSW2を正しく設定します。SW1のRC7とSW2のRC6をON側にし、その他をOFF側にセットしてください。

③ジャンパーソケットJ3とJ4を設定します。J3及びJ4いずれも下側の"USB UART"と書かれた方にソケットをセットします。

■パソコン側の準備

ターミナルソフトはmikroCに付属しています。このターミナルソフトを使用すると大変便利に通信が行えます。
mikroCのツールバーにある"USART Terminal"のアイコンをクリックします。

ターミナルソフトが起動します。

今回は、通信速度115200bps、1ストップビット、ノンパリティーの設定にします。通信ポートは、お使いの環境に合わせて設定します。仮想COMポートをお使いの場合には、デバイスマネージャで確認します。また、送信するデータはASCIIコードとし、データの最後はキャリッジリターン(0x0D)で終端します。すなわち、数字は文字列として送信するということになります。下図のように設定してください。

"Com Port"は、PICD-700SXと接続したRS232Cのポート番号を設定します。"Baud rate"は通信速度で、115200 bpsを選択します。

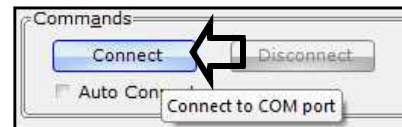
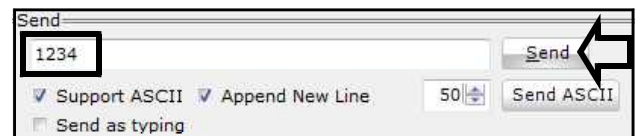
"Data Format"のところは"ASCII"にチェックを付け、その隣の"New Line Settings"では"CR(0x0D)"にチェックを入れます。

"Send"のところでは、"Support ASCII"と"Append New Line"の2つにチェックを入れます。

■コンパイルと書き込みの実行

これまでのチュートリアルと同様、ビルドしてHEXファイルをPICD-700SXのマイコンに書き込んでください。

■動作確認

1 Usart Terminalのソフトウエアの"Connect"ボタンを押して、RS232Cの接続を行います。

2 "Send"のウインドウのところに7セグメントLEDに表示したい数字を4桁で入力します。例えば、 1234 と入力してみましょう。

3 入力したら、右隣の"Send"ボタンをクリックします。

4 PICD-700SXの7セグメントLEDに 1234 と表示されていれば成功です。別の数字4桁を入力して色々な数字が表示されることを確認します。

■プログラムの解説

INTCON=0b11100000

割込用の設定レジスタINTCONに値を代入しています。TMR0と周辺機能の割込を有効にしています。
周辺機能についての詳細は更にPIE1レジスタで設定します。

OPTION_REG = 0b10000100;

TMR0の分周比などが決められるOPTION_REGレジスタの設定です。ここでは分周比を1:32に設定しています。
8MHz/4/32/256=244.140625Hzとなり4.096ミリ秒毎にTMR0がオーバーフローします。

PIE1 = 0b00100000;

周辺機能の割込についての設定です。ここではUART受信割込のみを有効に設定しています。

PIR1.B5 = 0;

PIR1のビット5はUART受信割込のフラグビットです。UART信号受信でここのビットが1になります。
このビットは割込ルーチンで0にクリアしてからmain関数に戻る必要があります。

UART1_Init(115200);

UART通信の初期化関数です。通信速度を引数として指定します。ここでは115200bpsに設定しました。

115.2kbpsを選択した理由は、UARTの通信速度がTMR0の割込間隔に対して十分に速くないと、表示に問題が出たりUART通信のデータの取りこぼしが出たりするためです。TMR0は約4ミリ秒毎の割込なので、それよりも十分速い115.2kbpsを選択しました。

if(INTCON.TMR0IF == 1){

割込ルーチン内での多重割込について分岐する部分です。今回はTMR0のオーバーフロー時とUART信号受信時に割込が発生します。どちらの割込発生で割込ルーチンに飛んできたのかを判定する必要があります。それには割込時に1に値が変化するフラグビットをif文で判定すればよいことになります。

ここではTMR0がオーバーフローするとINTCONレジスタのTMR0IFビットが1になるため、それを評価することにしました。

switch (num[digit]){

swich文です。( )内の値に応じてcase: で指定した部分に処理を分岐します。ここでは、num配列の値によって分岐し、PORTDに値を代入して7セグメントLEDの数字を形成しています。

変数digitは、桁指定の変数で0～3までの値を繰り返します。

cycle = cycle << 1;

ビット演算子の<<は左シフトです。シフトとは1の値を左にずらすこと(シフトすること)を示します。

値0000001を左に1つシフトすると、00000010となります。

7セグLEDのコモン端子の制御はRA0～RA3の各ピンを順番に1にしていくことに他なりませんので、左シフトを使って、それをPORTAレジスタに代入することでコモン端子の制御を行っています。

if(cycle>0b00001000) cycle=1;　で4桁目(RA3)まで1になったら、1を代入して再度RA0から順番に1にしています。

if(PIR1.RCIF == 1){

割込フラグビットの判定用if文です。UART受信割込発生でPIR1レジスタのRCIFレジスタが1になるため、それを判定しています。

for(i=3 ; i>=0 ; i--){

for文でnum配列の[0]～[3]にUARTで受信した1バイトの文字列を代入しています。変数iの値が3から0になるまで、iの値を1ずつデクリメントしていきます。 ">="は以上を示します。

while(!UART1_Data_Ready());

UART受信でデータが受信されるのを待機します。

データが受信されると、UART1_Data_Ready()関数は1を返します。0の間は、同じ場所をループしたいのでこの関数の直前に!演算子を付けて、論理を反転させて、戻り値が0の時while文としてループを繰り返すようにしています。

num[i] = uart_data-0x30;

UARTで受信したデータから0x30を減算して数字格納用の配列に代入しています。

パソコン側から送信されてくるデータはASCIIコードです。例えば数字文字の"1"は、ASCIIコードで0x31、同様にして例えば文字"9"は、0x39です。よってASCIIコードが数字であれば、0x30を引けば、値にすることができます。

if (uart_data == 0x0D)break;

キャリッジリターン(0x0D)を受信したらfor文から抜け出す記述です。break文はループから抜け出したり、条件文から抜け出す時に使用します。

INTCON = 0xE0;
PIR1.B5 = 0;
RCSTA.B4 = 0;
RCSTA.B4 = 1;
TMR0 = 0;

割込ルーチンを完了する前に各レジスタの値を初期化します。これを忘れたり設定が不適切な場合、割込が発生しなくなったり、意図しない時に割込が発生したりプログラムの不具合につながります。

まずINTCONの値は初期設定の値に戻しておきます。PIR1.B5はUART受信割込のフラグビットですので、0に戻しておきます。

RCSTA.B4は継続受信を可能にするCRENビットです。UARTのデータ受信をするRCREGレジスタは2つあるのですが、2つとも読み出されていない場合にはRCSTAレジスタのOERRビットが"1"になり、オーバーランエラーが発生したことが表示されこれ以降の受信動作は行われなくなります。この状態を正常に戻すのは一度RCSTAレジスタのCRENビットをクリアし、再びセットします。

最後にTMR0の値を0に戻しています。

このプログラムではmain関数は大変シンプルです。初期化の関数を呼び出すだけです。つまりまだまだ別の処理をmain関数内に記述できることを意味しています。PICマイコンのようなシングルタスクのCPUでは、複数の処理を同時に行うことは大変難しいのですが、割り込み処理を使うと、比較的効率よくプログラムを書くことができます。逆に、7セグメントLEDのコモン処理のように逐次処理をしなくてはいけないプログラムの場合、割込を使わないと他の処理ができなくなってしまいます。

チュートリアル⑤　～ICD機能を使ったデバッグを体験する～

■ICD機能とは？

ICDとはインサーキットデバッグの略で、ターゲットボード上のデバイスに実際にプログラムを書き込み、動作させながらパソコン上でデバッグを行う手法のことです。

例えば現在実行している行をハイライト表示させながら実機での動作が見られますので、プログラムの挙動を簡単に把握できます。また、1行ずつプログラムを実行したり、変数やレジスタの現在の値をリアルタイムに閲覧することができたりします。

本セットに付属のmikroCにはPICD-700SXと組み合わせて、C言語レベルでデバッグのできるICD機能が搭載されています。ICD機能を体験してみましょう。

■ICD機能を使うには

ICD機能を使用する場合、プログラムには特に大きな変更点はありませんが、delay_(ms)などのdelay関数が入ったプログラムは正しく動作しません。なぜならば、delay関数は、1命令を実行するほんの少しのクロックの時間を何周も実行させて目的の時間を作っています。そのため、delay関数までも1ステップ実行させてしまうと、膨大な繰り返しを実行せねばならず、ICD機能として実用的ではなくなってしまうからです。よって、ICDを使用する場合には、delay関数をプログラム中からなくして実行してください。また、関数によっては、1つの関数で大量の処理を実行するため、ステップ実行（1行ずつ実行）では、時間がかかりすぎる場合があります。

ICD機能を使用する場合には、コンパイル時に"Build type"を"ICD debug"設定にする必要があります。このチュートリアルを通してICDの使い方を一通り体験してみましょう。

■プログラムの内容

LCDに文字"microtechnica"を表示されます。
但しICD機能の動作を確認するため1文字ずつ表示するプログラムを作ります。プロジェクトの作り方などは今までのチュートリアルと全く同じです。

なお、このプログラムでは付属しているPIC18F45K22を8MHzで動作させて試すことにします。

■PICの交換

PICマイコンを交換します。ボードからPIC16F1939を取り外します。その後、PIC18F45K22を取り付けてください。装着方向に注意してください。

なお発振子はそのまま8MHzのクリスタルを使用しますので、ボードの"OSC1"に8MHzの水晶発振子が取り付けられていることを確認してください。

■プログラムの作成

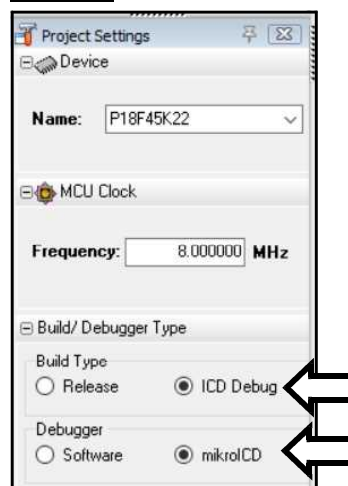
新しいプロジェクトを作成して、プログラムを作ってください。
使用するPICマイコンはPIC18F45K22に、動作周波数は8MHzとしてください。

次のプログラムを記述してください。

```
sbit LCD_RS at RB4_bit;
sbit LCD_EN at RB5_bit;
sbit LCD_D4 at RB0_bit;
sbit LCD_D5 at RB1_bit;
sbit LCD_D6 at RB2_bit;
sbit LCD_D7 at RB3_bit;
sbit LCD_RS_Direction at TRISB4_bit;
sbit LCD_EN_Direction at TRISB5_bit;
sbit LCD_D4_Direction at TRISB0_bit;
sbit LCD_D5_Direction at TRISB1_bit;
sbit LCD_D6_Direction at TRISB2_bit;
sbit LCD_D7_Direction at TRISB3_bit;

char text[14]="microtechnica";
int i;

void init(){
    ANSELA = 0;
    ANSELB = 0;
    TRISB = 0;
    Lcd_Init();
    Lcd_Cmd(_LCD_FIRST_ROW);
    Lcd_Cmd(_LCD_CURSOR_OFF);
    Lcd_Cmd(_LCD_CLEAR);
    delay_ms(100);

}

void main(){

    init();

    for(i=0 ; i<13 ; i++){
        Lcd_Chr_Cp(text[i]);
    }

}
```

■ビルドタイプの設定

プログラムが書き終わったら、ビルドタイプを指定します。このとき、従来とは違い、Build typeをICD用に変更します。

画面左側のサイドメニューの"Project Settings"にある"Build/Debugger type"のチェックボックスの、"ICD debug"と"mikroICD"にチェックを入れます。

■コンフィギュレーションビットの設定

　コンフィギュレーションビットの設定を行います。メニューバーの"Project"→"Edit project"で設定画面を表示させせます。

　必ず最初に1回"Default"ボタンを押して標準設定にします。続いて次の項目を下記のように変更します。

・Oscillator Selection	HS Oscillator
・4X PLL Enable	Disabled
・PORTB A/D	Disabled
・MCLR Pin	MCLR pin enabled, RE3..
・Background Debug	Enabled
・Low voltage Program	Disabled

　その他の項目は、Defaultボタンを押した後の標準設定で結構です。上記の設定は特に注意してご確認ください。

■ICDを動作させてみる

　ICDを実行して、プログラムの動作内容を確認しましょう。
LCDに何も文字列が表示が出ていないことを確認します。文字列が表示されている場合には一度PICD-700SXの電源を切断してLCDの画面をクリアした状態で下記を実行してください。

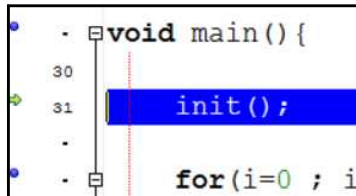
1　メニューバーの"Run"→"Start Debugger"をクリックします。
　キーボードの"F9"キーでも同様の操作ができます。

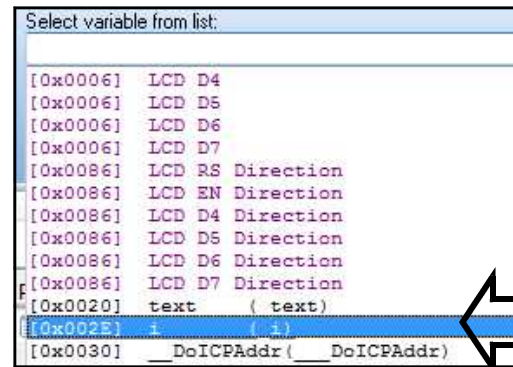
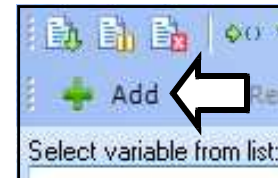
2　void main(){ の行が青色バーでハイライトされます。ICDでは実行される行が青色バーでハイライト表示されます。

　※void main(){の行が青くハイライトされない場合には、もう一度F9キーを押すか、"Run"→"Start Debugger"をクリックします。

　※void main(){の行が青くハイライトされず、別のアセンブラの画面が表示されてしまった場合には、メニューバーの"Run"→"Disassembly mode"をクリックして、画面をC言語の画面に戻してください。

3　この状態で1行ずつ実行するステップ実行を行い、1文字ずつLCDに文字が表示されるのを確認しましょう。

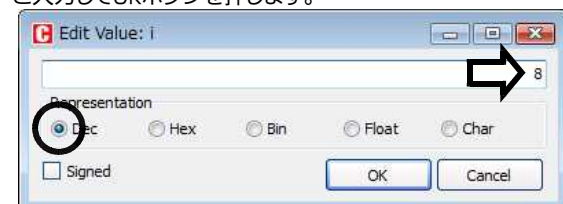
　キーボードのF8キーを1回押します。又はメニューバーの"Run"→"Step Over"をクリックします。F8キーを押すと、青色バーでハイライト表示された部分が移動します。
　F8キーを1回押すと、"init( )"の行にハイライトがジャンプします。

4　再度F8キーを押すと、ハイライトがfor文の中に入ります。
　何回かF8キーを押すと、プログラムはfor文を繰り返し実行して、1文字ずつLCDに文字が表示されていきます。5文字程度表示されたら次の手順を実行します。

5　ICDを実行すると、ウインドウの右側にWatch Windowが表示されます。ウォッチウインドウは、変数やレジスタの値をリアルタイムで閲覧できるウインドウです。ここでは変数iの値が1ずつ増えていく様子を確認しましょう。
　変数iを追加します。"Select Variable form list"のプルダウンをクリックして、その中から
「[0x---] i　(_i)」と表示されている部分クリックします。

6　Addボタンを押して、追加します。なお、リストから見つけにくい場合には、検索することもできます。
　その場合には、"Search for variable by assembly name"の部分に _i と入力します。変数名は、_(アンダーバー)に続けて変数名を入力します。

7　リストに追加されるとNameとValue、Addressが表示されます。Valueの値は現在の変数iに格納されている数値です。F8キーを押してFor文を実行すると、1ずつiの値が増えていく様子を見ることができます。

　値が更新されると数値は赤文字で表示されます。

8　この値はユーザーが自由に変更することができます。
　変数iは、text配列のインデックス値となっていますので、変数iの値を変えるということは、表示する文字を変えると言うことになります。ここでは試しに変数iの値を8にしてみましょう。

　変数iのValueの値をクリックします。すると、横に四角いボタンが表示されますのでクリックします。

　Edit Value: i というダイアログが表示されますので、"Representationの部分から10進数を意味する"Dec"をクリックします。値を8と入力してOKボタンを押します。

9 変数iの値を強制的に8に変えましたので表示される文字列も変わるはずです。キーボードのF8キーを4回押して実行してみましょう。文字が変わることを確認します。

10 ICD機能を終了する場合にはメニューバーの"Run"→"Stop Debugger"をクリックします。

■ICD機能を使用しない場合の注意

ICD機能を使用する場合、コンパイル時のBuild typeをICD用に設定しました。このICD用に設定されて生成されたHEXファイルは、通常動作はしません。ICD機能を使用しない通常のHEXファイルをビルドしたい場合には必ずビルドする前に"Build type"から"Release"にチェックを入れて、ビルドを行ってください。

mikroCのサポートフォーラム

mikroCは、世界中で使用されているユーザーの大変多いCコンパイラーです。mikroCの開発元であるmikroElektronika社では、サポートフォーラムを開設しており、活発な意見交換が行われています。

言語は英語ですが、検索をすることで色々な手法や問題の解決策が掲載されていますので、ぜひご利用ください。

http://www.mikroe.com/forum/

なお、記事の閲覧だけであれば登録は必要ありませんが、投稿する場合にはメンバー登録が必要になります。

mikroC製品版へのアップグレード

PICD-700SXに付属のmikroCはコードサイズが2KBに制限されている体験版です。機能制限はありませんので、C言語の学習やチュートリアルを行うぶんには問題ありませんが、本格的なプログラムを記述する場合には容量が足りなくなります。体験版で2KB以上のコードを生成しようとすると、"Demo Limit"という表示がでてコンパイルができません。その場合には、製品版をご購入頂く必要があります。

PICD-700SXをお買い上げのお客様は特別にmikroC製品版を通常販売価格の10%引きにて販売させて頂きます。ご注文の際には下記のページよりお申し込み頂けます。その際クーポンコードの入力欄に下記のコードを入力ください。10%引き価格が適用されます。

http://www.microtechnica.tv/script/mikroC_reg/mikroC_reg.html

使い方で困った時は?

エラーが表示される場合などは、インターネットの検索エンジンでそのエラーメッセージ文で検索してみてください。ほとんどの場合、その解決方法が書かれたページがヒットします。当方のサイトを始め、本製品の開発元が開設しているサポートフォーラム(英語)などに投稿があればそれらのスレッドがヒットします。

ほとんどのエラーメッセージなどはここで探すことができますので、エラーメッセージが出て困った・・という場合には是非お試しください。

その他過去、当方にお客様から頂いたご質問について一般的な内容については当方のFAQページで解決方法などをご案内しています。こちらでも既知の問題はほとんどが掲載されていますので、是非ご覧ください。

http://www.microtechnica.tv/faq/faq.cgi

上記ページの「PIC統合評価ボード[PICD-700SX] FAQ」の項目や「mikroC Cコンパイラー FAQ」の項目をご参照ください。

mikroC PRO for PIC 10%割引クーポンコード

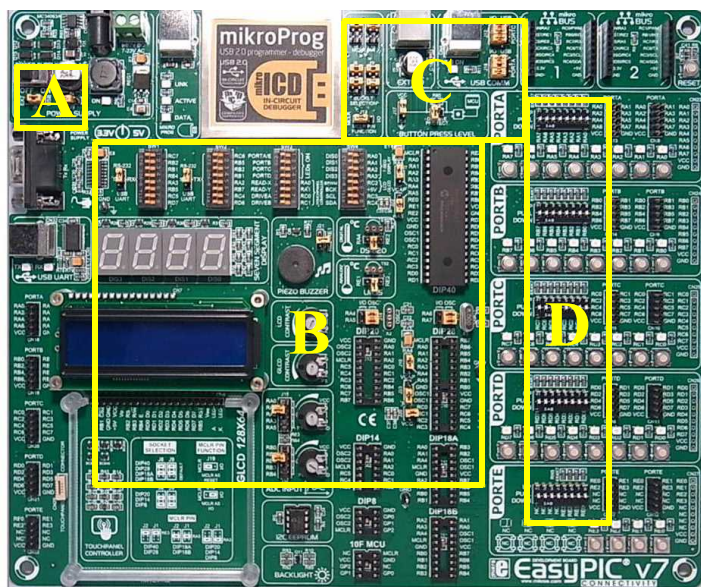
※有効期限: 製品ご購入後から1年間有効

PICD-700SXをお買い上げ頂きましたお客様は、mikroC PRO for PIC製品版の「ファイルライセンス版」を割引価格(表示価格の10%引き)にてご提供させて頂きます。製品版にして頂くと、生成されるHEXファイルのコードサイズが2kB以上でも可能となりフル機能がお使い頂けます。下記のサイトからご注文頂けます。

http://www.microtechnica.tv/script/mikroC_reg/mikroC_reg.html

※USBドングルタイプのライセンスは割引対象外です。

PICD-700SXのDIPスイッチ、ジャンパーソケット位置デフォルトポジション

PICD−700SXにはDIPスイッチやジャンパーソケットが多数実装されています。これらの工場出荷時の状態は下図のようになっております。
動作に問題がある場合や、ボードの設定を工場出荷時の状態に戻したい場合には下図を参考に、デフォルトポジションに設定を戻してお試し下さい。

※「開放」表示は、ソケットを装着せずピンをオープンにすることを意味します。

■ブロックA

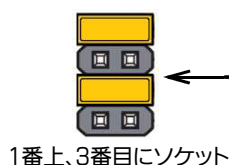
部品番号	標準位置	概　要
J5	5V側	供給電圧設定
J6	USB側	本体給電方法設定

■ブロックB

部品番号	標準位置	概　要
SW1	全OFF	UART接続先設定
SW2	全OFF	UART接続先設定
SW3	全OFF	LED/タッチパネルドライバ接続設定
SW4	全OFF	7セグLEDコモン/LCDバックライト/I2C接続設定
J3	RS232側	RXピン接続先設定
J4	RS232側	TXピン接続先設定
J7	RA4側	Vcap設定
J22	RA5側	Vcap設定
J11	開放	DS18S20信号線接続先設定
J25	開放	LM35信号線接続先設定
J13	OSC側	発振子接続設定
J14	OSC側	発振子接続設定
J10	RA4側	Vcap設定
J23	RA5側	Vcap設定
J20	開放	RA5ピンVcc接続設定
J15	開放	ADC用可変電圧印加ピン設定
J16	開放	ADC用可変電圧印加ピン設定
DIP40	PIC	PIC16F1939装着

■ブロックC

部品番号	標準位置	概　要
J1	最も上	MCLRピン設定(def.40ピン用設定)
J2	最も上	MCLRピン設定(def.40ピン用設定)
J8	左図参考	ICソケット選択設定
J9	左図参考	ICソケット選択設定
J19	MCLR側	MCLRピンの機能設定(def.Reset)
J17	GND側(下)	タクトスイッチ押下時接続先設定
J24	開放	短絡回路防止抵抗迂回路設定
J12	I/O側(左)	USB信号線接続先設定
J18	I/O側(左)	USB信号線接続先設定

1番上、3番目にソケット

■ブロックD

部品番号	標準位置	概　要
PORTA	中間位置	ピン別Pull−up/down設定
PORTB	中間位置	ピン別Pull−up/down設定
PORTC	中間位置	ピン別Pull−up/down設定
PORTD	中間位置	ピン別Pull−up/down設定
PORTE	中間位置	ピン別Pull−up/down設定